Titre: Title:	Un langage de modélisation de domaines de travail quantitatifs pour la conception d'interfaces écologiques
Auteur: Author:	Alexandre Moïse
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Moïse, A. (2009). Un langage de modélisation de domaines de travail quantitatifs pour la conception d'interfaces écologiques [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8462/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8462/
Directeurs de recherche: Advisors:	Jean-Marc Robert, & Marie-Hélène Noiseux
Programme: Program:	Génie industriel

UNIVERSITÉ DE MONTRÉAL

UN LANGAGE DE MODÉLISATION DE DOMAINES DE TRAVAIL QUANTITATIFS POUR LA CONCEPTION D'INTERFACES ÉCOLOGIQUES

ALEXANDRE MOÏSE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)
(GÉNIE INDUSTRIEL)
AVRIL 2009

Your file Votre référence
ISBN: 978-0-494-53803-6
Our file Notre référence
ISBN: 978-0-494-53803-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

UN LANGAGE DE MODÉLISATION DE DOMAINES DE TRAVAIL QUANTITATIFS POUR LA CONCEPTION D'INTERFACES ÉCOLOGIQUES

présentée par : MOÏSE Alexandre

en vue de l'obtention du diplôme de : Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. TRÉPANIER, Martin, ing., Ph.D., président

M. ROBERT, Jean-Marc, Doctorat, membre et directeur de recherche

Mme NOISEUX, Marie Hélène, Ph.D., membre et codirectrice de recherche

M. GERBÉ, Olivier, Ph.D., membre

M. DAVID, Bertrand, Doctorat, membre externe

REMERCIEMENTS

La réussite d'une thèse est le fruit d'une aventure à laquelle participent non seulement le thésard, mais également plusieurs personnes qui l'entourent. Sans elles, on peut dire qu'il est presqu'impossible d'atteindre le fil d'arrivée. Du fond de mon cœur, je vous remercie chaleureusement!

Jean-Marc Robert, professeur titulaire au département de mathématiques et de génie industriel de l'École Polytechnique de Montréal, et Marie Hélène Noiseux, professeure titulaire au département de finance de l'École des sciences de la gestion de l'Université du Québec à Montréal (UQÀM). Ces deux personnes ont su pousser les limites de mes capacités en combinant leurs expertises respectives.

Pour leur soutien financier, Fonds Nature et Technologie du Québec, Systèmes financiers Nortek et la Chaire en management des services financiers de l'UQÀM.

Les employés de Systèmes financiers Nortek, Gestion de patrimoine Tandem et Finalb pour avoir partagé leur savoir-faire.

Denis Paradis, Linda Pépin et Carmen Bernier de HEC Montréal qui ont cru en mes capacités d'enseignant et qui m'ont épaulé financièrement en m'offrant plusieurs charges de cours au fil des années.

Les collègues au doctorat et à la maîtrise de l'École Polytechnique de Montréal, particulièrement Eric Brunelle, Hélène L'Heureux et Lévis Thériault avec qui j'ai eu l'occasion de discuter longuement sur des sujets divers rattachés de près ou de loin à mes recherches (et les leurs).

Janik, Ambre et Virginie. Trois personnes qui ont successivement partagé ma vie durant ces longues années et qui ont dû composer avec la vie d'un thésard.

Les membres de mon jury, Bertrand David, Olivier Gerbé et Martin Trépanier, pour leur commentaires instructifs à l'égard de cette thèse et bien plus.

Ma famille et mes amis qui se sont intéressés à mon sujet de recherche (sans trop le comprendre) et qui m'ont questionné pour savoir où j'en étais rendu, quand j'allais terminer, ce que j'allais faire après... Et de me rappeler constamment de ne pas lâcher et de les inviter à la soutenance!

Finalement, mes parents, France et Jean-Claude qui ont fait de moi ce que je suis aujourd'hui.

RÉSUMÉ

Une interface écologique (IE) est un type particulier d'IU permettant à son utilisateur de s'adapter aux événements imprévus en présentant de manière explicite la hiérarchie fonctionnelle du domaine de travail, c'est-à-dire l'ensemble des contraintes inter-reliées qui régissent le domaine de travail. Concrètement, une IE intègre un ensemble de composants visuels, c'est-à-dire un agencement de formes géométriques, dont chacun correspond à une de ces contraintes. Afin d'aider le concepteur d'IE, la littérature présente un thésaurus visuel. Il s'agit d'un catalogue de composants visuels réutilisables. Ainsi, le concepteur d'IE n'a pas à concevoir de composants visuels pour chaque nouvelle IE, mais plutôt à sélectionner du catalogue les composants visuels appropriés pour ensuite les disposer sur une surface d'affichage.

Puisque les composants visuels représentent graphiquement des contraintes du domaine de travail, il devrait exister une correspondance directe entre ces derniers et leur définition lors de l'analyse. Il n'en est toutefois pas ainsi. Dans un contexte de conception d'IE, la littérature présente la hiérarchie d'abstraction et de décomposition (HAD) comme étant la seule technique de représentation de domaines de travail. Or, la conception d'IE nécessite plus que ce qui est représenté par la HAD; celle-ci doit être accompagnée d'une liste des variables et d'une liste des équations. La HAD offre uniquement une vue de haut niveau de la hiérarchie fonctionnelle. Quant à la liste des variables et la liste des équations, en s'appuyant sur la HAD, elles servent à définir les contraintes du domaine de travail qui composent la hiérarchie fonctionnelle. Le problème est que la définition des contraintes du domaine de travail sous forme d'une liste d'équations ne s'appuie pas sur la même structure que celle sous-jacente aux composants visuels. Dans le premier cas, les contraintes sont écrites librement sous la

forme d'équations mathématiques. Dans le deuxième cas, les composants visuels s'appuient sur une structure précise de relations entre termes et opérateurs.

Cette différence de structures pose un obstacle dans l'utilisation du thésaurus visuel en ce sens que le concepteur d'IE doit redéfinir les contraintes définies initialement lors de l'analyse du domaine de travail afin qu'elles correspondent à la structure des composants visuels qui font partie du catalogue. Afin de solutionner ce problème, cette thèse propose un nouveau langage de modélisation d'analyse de domaines de travail, appelé WoDoMoLEID, qui s'appuie sur le méta-modèle des IE. Son objectif est double. D'une part, ce dernier doit permettre de définir de manière intégrée toutes les informations essentielles à la conception d'IE, c'est-à-dire les objets, les contraintes et la hiérarchie fonctionnelle d'un domaine de travail. D'autre part, grâce à sa structure, celui-ci doit permettre une correspondance directe avec des composants visuels réutilisables. Ce dernier point constitue le premier pas vers l'automatisation du processus de sélection des composants visuels réutilisables.

Afin d'évaluer le langage de modélisation proposé, celui-ci a été appliqué au domaine de la gestion de portefeuille d'actifs financiers. De plus, un prototype de logiciel de modélisation a été développé et un catalogue de composants visuels réutilisables a été créé. Les résultats démontrent que WoDoMoLEID permet d'atteindre les deux objectifs mentionnés précédemment.

ABSTRACT

An ecological interface (EI) is a particular type of UI that allows its user to adapt to unforeseen events by explicitly displaying the work domain functional hierarchy, that is, all the interrelated constraints governing the work domain. Concretely, an EI incorporates a set of visual components, that is, a combination of geometric shapes, each corresponding to one of these constraints. In order to help the EI designer, the literature presents a visual thesaurus. It is a catalogue of reusable visual components. Thus, the EI designer does not have to design each individual visual component for a new EI, but needs only to select the appropriate visual components from the catalogue and organize them on a display area.

Because the visual components represent graphically the work domain constraints, there should be a direct correspondence between them and their definition during analysis. It is not the case. In an EI design context, the literature presents the abstraction and decomposition hierarchy (ADH) as the only work domain representation. However, EI design requires more than what is represented by the ADH. That is why it needs to be completed by a list of variables and a list of equations. The ADH is only a high-level view of the functional hierarchy. As for the list of variables and the list of equations, both based on the ADH, they are used to define the work domain constraints that compose the functional hierarchy. The problem is that the definition of the work domain constraints as a list of equations is not based on the same structure as the one the visual components are based on. In the first case, the constraints are freely written in the form of mathematical equations. In the second case, the visual components are based on a precise structure of relationships between terms and operators.

This difference in structures constitutes a barrier in using the visual thesaurus in the sense that the EI designer needs to redefine the constraints initially defined during work domain analysis to match the structure of the visual components that are part of the catalogue. To solve this problem, this thesis proposes a new modeling language for work domain analysis, called WoDoMoLEID, based on the meta-model of EI's. Its objective is twofold. On the one hand, it should allow defining in an integrated manner all information essential to the design of EI's, that is, objects, constraints and the functional hierarchy of a work domain. On the other hand, because of its structure, it it should allow a direct correspondence with reusable visual components. This last point is the first step towards automating the process of selecting reusable visual components.

In order to evaluate the proposed modeling language, it was applied to the financial assets portfolio management work domain. In addition, a modeling software prototype has been developed and a catalogue of reusable visual components has been created. The results show that WoDoMoLEID can achieve both aforementioned objectives.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGNES ET ABRÉVIATIONS

CIE	Conception d'interfaces écologiques
HAD	Hiérarchie d'abstraction et de décomposition
IDE	Ingénierie dirigée par les modèles
IE	Interface écologique
IU	Interface utilisateur
KBB	Knowledge-Based Behaviour
L1G	Langage de première génération
L2G	Langage de deuxième génération
L3G	Langage de troisième génération
L4G	Langage de quatrième génération
LMG	Langage de modélisation générique
LMSD	Langage de modélisation spécifique au domaine
OCL	Object Constraints Language
OMG	Object Management Group
OQLF	Office québécois de la langue française
RBB	Rule-Based Behaviour
SBB	Skill-Based Behaviour
STI	Système de traitement d'informations
UML	Unified Modeling Language
WoDoMoLEID	Work Domain Modeling Language for Ecological Interface Design

« Productive work in today's society and economy
is work that applies vision, knowledge and concepts—
work that is based on the mind rather than on the hand. »
(Drucker 1959, p.120)

« People propose, Science studies, Technology conforms »
(Norman 1993)

« In designing decision-making organizations,
we must understand not only the structure of the decisions to be
made, but also the decision-making tools at our disposal,
both human and mechanical—men and computers. »
(Simon 1997, p.243)

« We cannot change the human condition,
but we can change the conditions under which people work. »
(Reason 2000, p.25)

CHAPITRE 1 - INTRODUCTION

Un système sociotechnique complexe est caractérisé par un certain nombre de facteurs notamment sa nature dynamique, un nombre élevé de personnes qui interagissent les unes avec les autres, plusieurs sous-systèmes inter-reliés, une automatisation des traitements, un contrôle réalisé par le biais d'ordinateurs ainsi qu'un haut niveau de risque et d'incertitude (Vicente 1999a)[1]. Il s'agit donc d'un système ouvert assujetti à des perturbations imprévues qui, par définition, ne peuvent être évitées et peuvent être accompagnées de conséquences désastreuses lorsqu'elles surviennent (Vicente et Rasmussen 1992). On peut penser à un portefeuille d'actifs financiers, à une entreprise, à un réacteur nucléaire, à un avion ou au corps humain. Ainsi, la bonne conduite d'un système sociotechnique complexe est dépendante de la capacité d'adaptation de la personne qui est responsable de son contrôle.

La conception d'environnements de travail soutenant l'adaptation exige d'obtenir une connaissance du domaine de travail, c'est-à-dire du système sociotechnique complexe. Cette thèse s'intéresse à la problématique de la représentation de ce dernier à des fins de conception d'interfaces utilisateurs (IU).

1.1. Mise en contexte

La conception centrée sur l'utilisateur, telle que définie par la norme ISO 13407, consiste à mettre l'utilisateur au centre de la démarche de création de systèmes[2]

---

[1] Pour qu'un système soit caractérisé comme étant un système sociotechnique complexe, il n'a pas à correspondre à tous ces facteurs, mais doit avoir une correspondance forte avec un certain nombre d'entre eux. En somme, un système sociotechnique complexe peut mettre l'accent sur les aspects sociaux et un autre peut mettre l'accent sur les aspects techniques. Pour une définition précise, voir Vicente (1999a) aux pages 5 ainsi que 14 à 17.

[2] Le grand dictionnaire terminologique de l'Office québécois de la langue française (OQLF, www.granddictionnaire.com) définit un système comme étant un « Ensemble d'éléments associés pour

interactifs. Une des étapes de cette démarche est la définition du contexte d'utilisation qui consiste à décrire la tâche, l'utilisateur et l'environnement dans lequel le système sera utilisé. Selon Maguire (2001), pour des systèmes familiers, des rencontres avec les parties prenantes sont suffisantes pour décrire le contexte d'utilisation. Toutefois, pour des systèmes moins familiers, qui peuvent s'avérer complexes, ces rencontres doivent être complétées par, entre autres, une analyse de tâches.

Diaper (2004, p. 15) explique que le travail consiste à réaliser des changements à un environnement. En adoptant une perspective d'interactions humain-ordinateur, il mentionne que le travail est réalisé par des individus par l'entremise de matériel informatique. Le travail d'un individu est guidé par des buts, c'est-à-dire des états désirés de l'environnement. Pour atteindre ses buts, un individu doit réaliser un ensemble de tâches qui consistent à modifier l'état de l'environnement. L'analyse de tâches est donc l'étude de la manière dont le travail est réalisé par les tâches, c'est-à-dire les actions qui modifient l'environnement.

L'analyse de tâches sous-entend, selon Vicente (1999b), que les évènements qui amènent les individus à réaliser des tâches doivent être préalablement identifiés. En d'autres mots, l'analyse de tâches est tout indiquée pour les évènements fréquents ainsi qu'occasionnels mais anticipés. Toutefois, pour ce qui est des événements imprévus, si l'IU qu'il utilise a été conçue à partir d'une analyse de tâches, l'individu ne peut s'appuyer sur celle-ci pour le soutenir dans son travail. En somme, une IU conçue à partir d'une analyse de tâches ne permet pas, ou du moins permet difficilement, à son utilisateur de s'adapter à une situation imprévue.

---

atteindre un but déterminé au moyen d'un fonctionnement spécifié. » Cette définition, provenant du domaine de la cybernétique, englobe non seulement les logiciels, mais tout appareil ou mélange de matériels et d'individus destiné à soutenir une tâche quelconque. C'est donc en ce sens que le terme « système » est employé tout au long de cette thèse.

Devant un système sociotechnique complexe, un individu qui fait face à un événement imprévu se retrouve en situation de résolution de problème. En d'autres mots, ce dernier doit s'adapter à la situation en établissant une séquence d'actions permettant d'atteindre ses buts puisqu'aucune n'a été préalablement définie. Si l'analyse de tâches ne permet pas de concevoir une IU soutenant convenablement l'individu dans cette situation, existe-t-il une autre approche qui le permet?

La littérature sur la résolution de problème indique que le facteur déterminant dans la résolution de problème par un individu est la représentation interne du domaine de travail (Newell et Simon 1972, Rasmussen 1985, Vicente et Rasmussen 1992). Vicente (1999a, p. 10) définit un domaine de travail comme étant le système sous contrôle indépendamment des gens qui en font partis, des mécanismes d'automatisation, des événements, des tâches, des buts ou des interfaces humain-ordinateur. Par conséquent, afin de soutenir le processus de résolution de problème d'un individu, une IU doit présenter de manière explicite le domaine de travail. La conception d'une telle IU exige au préalable de recourir à des techniques d'analyse mettant l'accent non pas sur la tâche, comme c'est la coutume en ergonomie, mais sur la description du domaine de travail. Néanmoins, il est important de mentionner que l'analyse du domaine de travail et l'analyse de tâches ne sont pas des approches concurrentes, mais plutôt complémentaires (Miller et Vicente 2001, Burns et Hajdukiewicz 2004, Hajdukiewicz et Vicente 2004).

Vicente (1999b) présente trois critères que doivent respecter toute technique d'analyse du domaine du travail : (a) indépendance des artéfacts, (b) indépendance des événements et (c) faire preuve de pertinence psychologique. À ce sujet, il indique que les techniques d'analyse de tâches sont habituellement pertinentes d'un point de vue psychologique puisqu'elles tiennent compte des buts à atteindre comparativement à, par exemple,

l'analyse de systèmes qui se concentre sur les mécanismes internes décrivant le fonctionnement d'un système. Les IU conçues à partir d'une analyse de systèmes sont en général déficientes sur le plan de l'utilisabilité car elles ne correspondent habituellement pas au modèle mental de son utilisateur (Norman 2002). Le plus grand point faible de l'analyse de tâches est, tel que mentionné précédemment, qu'elle est dépendante des événements. De plus, même s'il existe des exceptions toujours selon Vicente (1999b), la plupart des techniques d'analyse de tâches ne peuvent se dissocier des artéfacts utilisés dans la réalisation des tâches. Bien que la description des tâches génériques puisse rester indépendante des artéfacts utilisés, la description des tâches élémentaires implique nécessairement l'interaction avec ceux-ci, ce qui a pour effet d'influer le déroulement de la tâche.

La littérature présente quelques approches de conception d'IU permettant de soutenir la résolution de problème et respectant les trois critères de Vicente (1999b). Ces approches, notamment fonction-comportement-état[3] (Lin et Zhang 2004, 2005), la modélisation de flux à multi-niveaux (Lind 1994, Larssen 1996) et la conception d'interfaces écologiques[4] (CIE) (Vicente et Rasmussen 1992), proposent différentes techniques de représentation du domaine de travail. Cette thèse se limite toutefois à une seule approche, la CIE, qui est incontestablement celle ayant attiré le plus d'attention dans la littérature en ayant fait l'objet de plus de 150 publications (Burns et Hajdukiewicz 2004, p. xxvi). De plus, la CIE a été appliquée à plusieurs domaines, notamment le contrôle de processus industriel, l'aviation, la gestion de réseaux informatiques et la médecine (Vicente 2002, Burns et Hajdukiewicz 2004), contrairement aux deux autres approches qui ont été appliquées uniquement au contrôle de processus industriels.

---

[3] Traduction libre du terme anglais *Function-Bahavior-State*.

[4] La raison pour laquelle le terme « interface écologique » a été adopté est que ce type d'IU s'appuie sur les notions d'une école de pensée en psychologie appelée « psychologie écologique » (Burns et Hajdukiewicz 2004).

La CIE a été appliquée à des domaines de travail de nature quantitative et de nature qualitative. Toutefois, les applications aux domaines de travail quantitatifs ont largement prédominé la littérature sur le sujet. De plus, les écrits portant sur le détail de la démarche de transformation des résultats de l'analyse du domaine de travail en une IE ont porté presqu'exclusivement sur des domaines de travail quantitatifs (Burns et Hajdukiewicz 2004). C'est pout cette raison que cette thèse se limite à la représentation de ce type de domaine de travail.

1.2. Problématique

La CIE est un cadre de travail théorique permettant de créer des IU soutenant la résolution de problème en rendant explicite le domaine de travail à travers celle-ci (Vicente et Rasmussen 1992, Vicente 1999a, Burns et Hajdukiewicz 2004). Plus concrètement, une interface écologique (IE) affiche un ensemble de composants visuels. Chaque composant visuel est constitué d'un agencement particulier de formes géométriques qui représente une contrainte d'un domaine de travail. Un domaine de travail est composé d'un ensemble de contraintes inter-reliées. Cette structure de contraintes constitue la hiérarchie fonctionnelle de celui-ci. En somme, une IE présente de manière explicite la hiérarchie fonctionnelle d'un domaine de travail[5].

En s'inspirant de Burns et Hajdukiewicz (2004), la figure 1.1 illustre de manière simplifiée le processus de création d'une interface écologique (IE) en mettant l'accent sur les activités et les produits du processus.

[5] Les IE sont décrites en détails au chapitre 3.

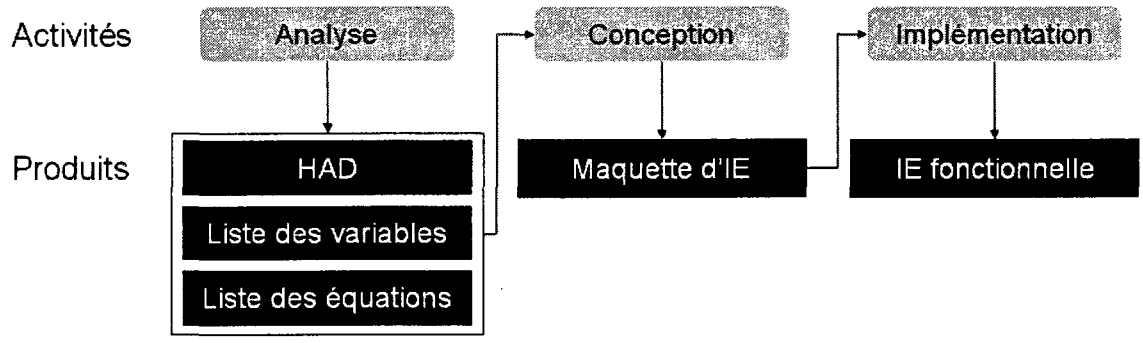


Figure 1.1 – Représentation simplifiée du processus de création d'une interface écologique fonctionnelle en s'inspirant de Burns et Hajdukiewicz (2004).

Le travail du concepteur d'IE consiste principalement à représenter la hiérarchie fonctionnelle du domaine de travail par des agencements de formes géométriques dont certaines pourront être directement manipulées par l'utilisateur (Burns et Hajdukiewicz 2004). Le résultat de ce travail est généralement une maquette de l'IE. Cette dernière est ensuite implémentée par un programmeur pour obtenir une IE fonctionnelle.

Plusieurs activités d'analyse peuvent précéder ce travail de CIE, par exemple l'analyse de tâches, de l'environnement physique d'utilisation et de la population d'utilisateurs. Quoi qu'il en soit, l'activité d'analyse la plus essentielle à la CIE est celle qui consiste à décrire le domaine de travail. À cette fin, la littérature propose la hiérarchie d'abstraction et de décomposition (HAD) comme étant la seule technique de représentation de celui-ci (Rasmussen 1983, Rasmussen 1985, Ramussen, Pejtersen et Schmidt 1990, Vicente et Rasmussen 1992, Vicente 1999a, Burns et Hajdukiewicz 2004).

À l'origine, la HAD, qui a vu le jour dans les années 1970, n'était pas destinée à concevoir des IE. Il s'agit d'une technique de représentation de domaines de travail compatible avec la manière dont un système complexe est représenté cognitivement par un individu en situation de résolution de problème (Rasmussen 1985, Vicente 1999a, Burns et Hajdukiewicz 2004). Son objectif est de représenter un domaine de travail,

comme son nom l'indique, en différents niveaux d'abstraction et de décomposition[6]. Cette représentation a été créée en s'appuyant sur les résultats d'études empiriques réalisées dans les domaines du dépannage électronique, bibliothécaire et des centrales nucléaires (Vicente 1999a, p. 364, Burns et Hajdukiewicz 2004, p. 9).

En somme, la HAD n'est autre qu'une technique de représentation d'opportunité pour la CIE; à défaut d'avoir une meilleure technique de représentation de domaines de travail, la HAD a été adoptée pour la CIE. Bien que cette dernière et les IE partagent la même philosophie, leur structure respective semble présenter certaines incompatibilités. Notamment, pour concevoir une IE, la HAD ne semble pas être suffisante; elle doit être accompagnée de représentations secondaires, en l'occurrence, une liste de variables et une liste d'équations (Bisantz et Vicente 1994, Burns et Hajdukiewicz 2004).

1.3. Objectifs de recherche

Cette section présente les deux objectifs de recherche qui se rapportent à la CIE.

1.3.1. Objectif 1

Quel que soit le domaine d'application, la conception est une tâche de résolution de problème selon Simon (1996). Il indique qu'un artéfact est une interface entre l'environnement interne, c'est-à-dire les mécanismes de l'artéfact, et l'environnement externe, c'est-à-dire le contexte dans lequel l'artéfact sera utilisé. En somme, la description de l'environnement externe équivaut à la description d'un problème à résoudre. Qui utilisera l'artéfact? De quelle manière? Pour quelles raisons? Dans quelles conditions? Les réponses à ces questions et à d'autres constituent les exigences auxquelles l'artéfact devra se conformer. Quant à la solution, il s'agit pour le concepteur

[6] La HAD est décrite en détails au chapitre 4.

d'élaborer l'environnement interne de l'artéfact afin qu'il soit approprié à son environnement externe.

Simon (1996, p. 132) indique que résoudre un problème signifie simplement le représenter de manière à rendre la solution transparente. Les constats de la section précédente permettent de se questionner sur la transparence de la solution, en l'occurrence une interface écologique, lorsque le problème est présenté sous forme de HAD. Bien que la littérature présente quelques critiques à l'égard de la HAD comme technique de représentation de domaines de travail (Lind 1999, 2003), il semble que personne ne se soit penché sur les problèmes reliés à l'utilisation de la HAD pour la conception d'IE. Par conséquent, le premier objectif de cette thèse est le suivant :

O1 Faire le point sur la hiérarchie d'abstraction et de décomposition pour la conception d'interfaces écologiques.

Cet objectif passe par une évaluation de la compatibilité entre la HAD et les IE. Évaluer la compatibilité entre des techniques de représentation consiste à voir celles-ci comme étant chacune un langage de modélisation. Un langage est, selon le Nouveau Petit Robert édition 2007, un système de signes vocaux ou graphiques permettant l'expression de la pensée et la communication entre des individus. Ce système est composé d'éléments (vocabulaire) et de règles d'associations (grammaire) entre ces éléments. La HAD et la représentation d'une IE sous forme de maquette sont toutes deux des systèmes d'éléments inter-reliés. Ce sont, par conséquent, des langages de modélisation.

La structure d'un langage de modélisation s'exprime par un méta-modèle. Il s'agit d'un modèle du système qui définit ce langage. Un méta-modèle est généralement défini comme étant un modèle d'un langage de modélisation; c'est un modèle qui définit la structure du langage de modélisation. L'évaluation de la compatibilité entre deux

langages de modélisation est donc réalisée en établissant des correspondances entre les éléments de la structure d'un langage (méta-modèle source) et les éléments de la structure d'un autre langage (méta-modèle cible). Dans ce cas, le méta-modèle source est le méta-modèle de la HAD et le méta-modèle cible est le méta-modèle d'une IE. Par conséquent, en plus des résultats de l'analyse, l'atteinte du premier objectif apporte trois autres contributions : (a) le méta-modèle de la HAD, (b) le méta-modèle d'une IE et (c) une méthode d'évaluation de la compatibilité entre méta-modèles.

1.3.2. Objectif 2

Vicente (2002) mentionne qu'un des obstacles à l'adoption des IE est que le choix de l'apparence des composants visuels est plutôt un art qu'une science; la transformation d'une HAD en une IE exige une grande part de créativité. Cet énoncé n'est plus tout à fait véridique depuis la publication du thésaurus visuel de Burns et Hajdukiewicz (2004). Il s'agit d'un catalogue de composants visuels réutilisables associés à des types de contraintes, par exemple une addition ou une division, qui s'appuient sur une structure particulière de relations entre des termes et des opérateurs mathématiques. Ainsi, en principe, un concepteur d'IE peut, à partir de ce catalogue, sélectionner les composants visuels correspondant aux contraintes d'un domaine de travail. Le problème est que, lors de l'analyse du domaine de travail, la définition des contraintes ne suit pas de règles précises; il s'agit d'équations mathématiques écrites librement (Bisantz et Vicente 1994, Burns et Hajdukiewicz 2004). Par conséquent, il est difficile pour un concepteur d'établir une correspondance entre les contraintes issues de l'analyse et les composants visuels du thésaurus de Burns et Hajdukiewicz (2004). Dans cette situation, il est clair que la représentation des contraintes n'est pas propice à la sélection directe des composants visuels appropriés. Le concepteur doit donc, à chaque nouvelle création d'IE, redéfinir les contraintes du domaine de travail afin de les adapter à la structure imposée par le thésaurus visuel, ce qui alourdit considérablement son travail.

Afin d'accélérer le temps associé à la conception d'IE, il faudrait faire en sorte que les contraintes définies en analyse le soient à partir de la même structure que celle sous-jacente aux composants visuels du thésaurus de Burns et Hajdukiewicz (2004). Pour ce faire, il faut définir des règles précises de définition de contraintes lors de l'analyse du domaine de travail. Ainsi, une contrainte définie par un analyste peut être directement associée à un composant visuel particulier. Ce faisant, le concepteur n'aura pas à allouer de temps à redéfinir les contraintes afin de les rendre compatibles à la structure du thésaurus visuel. De plus, il sera en principe possible d'automatiser la sélection des composants visuels à partir de contraintes du domaine de travail, ce qui aura pour effet d'accélérer encore plus le processus de conception d'IE. Par conséquent, le deuxième objectif de cette thèse est le suivant :

O2 **Créer et évaluer un langage de modélisation de domaines de travail quantitatifs permettant une correspondance directe avec les composants visuels d'une interface écologique.**

1.4. Structure de la thèse

Le chapitre 1 constitue l'introduction de cette thèse. Il aborde la mise en contexte, la problématique et les objectifs de recherche.

Cette thèse porte sur la création et l'évaluation d'un langage de modélisation. Ainsi, le chapitre 2 consiste en une recension de la littérature sur la modélisation. Il s'agit principalement de distinguer les notions de modèles et de méta-modèles. De plus, l'importance de la méta-modélisation est expliquée.

Le chapitre 3 consiste en une recension de la littérature sur les IE. Les fondements de la résolution de problème en s'appuyant principalement sur la théorie de Newell et Simon

(1972) sont présentés en premier lieu. Ensuite, une description des IE est réalisée. Dernièrement, un méta-modèle d'une IE est présenté.

Le chapitre 4 présente une analyse et une critique de la HAD. Un méta-modèle de cette technique de représentation du domaine du travail est présenté. Sa compatibilité est ensuite évaluée avec le méta-modèle d'une IE présenté au chapitre 3.

Le chapitre 5 présente le langage de modélisation d'environnements quantitatifs proposé dans cette thèse. Il présente en premier lieu un méta-modèle décrivant la syntaxe abstraite du langage de modélisation. Il s'agit des éléments de ce dernier et les relations entre ceux-ci. En deuxième lieu, ce chapitre décrit la syntaxe concrète du langage de modélisation, c'est-à-dire la notation utilisée pour représenter les éléments de la syntaxe abstraite.

Le chapitre 6 présente l'évaluation du langage décrit au chapitre 5. Un prototype de logiciel de modélisation est développé à cette fin. Le langage est appliqué à un domaine de travail particulier, celui de la gestion de portefeuille d'actifs financiers.

Le chapitre 7 est la conclusion de cette thèse. Les limites, les contributions et les avenues futures de recherches y sont présentées.

CHAPITRE 2 - LA MODÉLISATION

On dit qu'une image vaut mille mots. Cet adage, habituellement utilisé pour les photos ou les peintures, s'applique aussi bien à la modélisation graphique. Ce chapitre vise à décrire ce en quoi consiste la modélisation. Pour ce faire, il est nécessaire en premier lieu de s'interroger sur ce qu'est un modèle. La section 2.2 présente une synthèse de définitions présentées dans la littérature. Un modèle s'appuie sur un langage de modélisation qui regroupe habituellement une notation et un ensemble de règles. Ce concept est décrit à la section 2.3. Différents modèles sont utilisés à différentes étapes du cycle de développement de logiciels. Les modèles d'analyse et les modèles de conception sont différenciés à la section 2.4. Enfin, la section 2.5 présente une introduction à l'ingénierie dirigée par les modèles, une approche de conception qui met les modèles au premier plan. La section 2.6 présente une synthèse de ce chapitre.

2.1. Qu'est-ce qu'un modèle?

Force est de constater que la définition de ce qu'est un modèle peut porter à confusion. En effet, le Nouveau Petit Robert édition 2007 ne présente pas moins de sept définitions. La septième et dernière définition, qui se rapporte aux domaines scientifiques, concerne particulièrement cette thèse : « Représentation simplifiée d'un processus, d'un système ». Cette définition introduit deux notions : représentation et simplification. La première notion sous-entend que le modèle se distingue de ce qui est modélisé. La deuxième notion sous-entend l'abstraction de détails. Ces deux notions sont approfondies dans les sous-sections suivantes.

Le concept de modèle sera dorénavant abordé dans une perspective de génie logiciel, domaine dans lequel la modélisation est bien ancrée. La raison derrière ce choix est que

cette thèse s'intéresse au domaine de l'ergonomie cognitive, plus spécifiquement à la conception d'interfaces utilisateurs. Les interfaces utilisateurs aujourd'hui sont presqu'uniquement associées à des systèmes informatiques; elles constituent ainsi un composant d'un logiciel.

2.1.1. Un modèle est une représentation

La notion de modèles ne date pas d'hier. En effet, Favre (2004) et Chen (1999) indiquent qu'elle remonte à plusieurs millénaires. Ces deux auteurs font le rapprochement entre les modèles utilisés aujourd'hui pour le développement de logiciels et le langage de communication utilisé en ancienne Égypte. Principalement, les hiéroglyphes s'appuyaient sur une représentation graphique de concepts à des fins de communication qui suivaient des règles bien définies. De plus, Favre (2004) réfère aux artéfacts de cette période qu'on retrouve dans les musées comme étant des modèles; il s'agit de représentations de personnages importants, de divinités ou d'évènements.

Kurtev (2005, p. 13) indique qu'un modèle est utilisé pour étudier indirectement un objet. Il ajoute que plusieurs obstacles peuvent empêcher l'étude directe de celui-ci, notamment, parce qu'il est inaccessible, parce que son accès direct nécessite un investissement trop élevé ou parce que l'objet n'existe pas. Il est alors essentiel de recourir à une représentation d'un objet pouvant être de nature physique ou conceptuelle. Dans le même ordre d'idée, Bézivin et Gerbé (2001) mentionnent qu'un modèle doit permettre de répondre à certaines questions à la place de l'objet modélisé.

Afin de décrire la relation entre un objet et son modèle, Hughes (1997) présente les trois activités[7] illustrées à la figure 2.1 :

[7] Dans son article, Hughes (1997) associe ces activités à l'étude de modèles théoriques en physique. Toutefois, Kurtev (2005, p. 13) les applique de manière générique à tout type de modèle.

- **Dénotation.** La dénotation consiste à faire la correspondance entre des éléments de l'objet modélisé et les éléments correspondant d'un modèle. Le modèle doit donc être une représentation fidèle de l'objet modélisé sans toutefois être une réplique exacte.
- **Démonstration.** La démonstration consiste à tirer des conclusions d'un modèle.
- **Interprétation.** L'interprétation consiste à faire la correspondance entre les éléments d'un modèle et les éléments correspondant de l'objet modélisé. L'interprétation peut également être vue comme étant la projection de la démonstration vers l'objet, ce que Hughes (1997) appelle l'adéquation empirique.

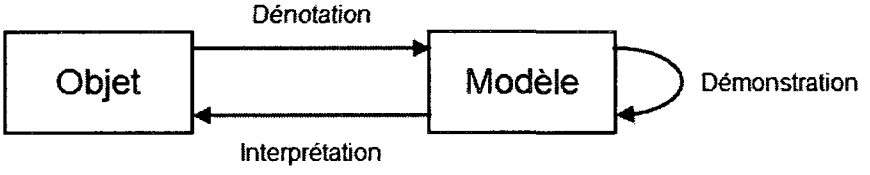


Figure 2.1 – Relations entre un objet et son modèle [adaptée et traduite de Hughes (1997)].

Un modèle se manifeste généralement sous deux formes ou une combinaison des deux :

- **Graphique.** Un modèle graphique utilise des symboles représentant des concepts d'un domaine particulier. Ceux-ci sont créés à partir de formes géométriques généralement reliées par des lignes afin de représenter leurs relations. Bien que la représentation puisse s'appuyer sur des techniques de visualisation bidimensionnelle ou tridimensionnelle, les symboles sont répartis sur une surface bidimensionnelle. Par exemple, le résultat d'une analyse hiérarchique de tâche ou la description de la structure d'un logiciel à l'aide d'un diagramme de classes sont des modèles graphiques.

- **Textuelle.** Un modèle textuel utilise des mots appartenant à une langue écrite. Les mots sont créés à partir de symboles appelés des lettres, mais contrairement à la forme graphique, ces symboles ne sont pas des éléments modélisés et ne sont donc pas utilisés pour la dénotation et l'interprétation. Les mots sont utilisés à cette fin. Habituellement standardisés, ces derniers sont associés à des concepts d'un domaine particulier. Par exemple, la description d'un scénario de cas d'utilisation ou le code source d'un logiciel sont des modèles textuels. Dans le premier cas, les mots ne sont pas assujettis à une standardisation; le modèle est décrit sous forme de prose. Dans le deuxième cas, chaque mot employé doit pouvoir être interprété par un compilateur. Par conséquent, le code source doit suivre des règles d'écriture très rigoureuses.

Pour quelles raisons une forme de représentation serait préférable à l'autre? Afin de répondre à cette question, il importe de distinguer le traitement cognitif des images et des mots. À cette fin, Paivio (2007) présente la théorie du double codage. Cette dernière postule que les informations visuelles et verbales sont traitées différemment par le cerveau puisque chaque type est assujetti à une représentation mentale distincte : les *imagènes*, associées à la représentation visuelle, et les *logogènes*, associées à la représentation verbale. La figure 2.2 illustre le traitement cognitif d'informations selon cette théorie. Puisque la tâche de modéliser est principalement de nature visuelle, les explications subséquentes portent exclusivement sur la vue même si cette théorie s'applique aux autres sens.

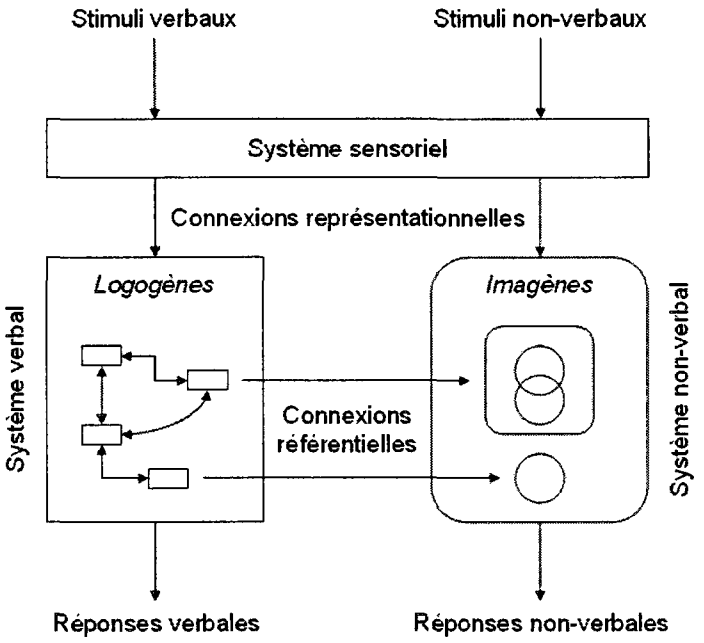


Figure 2.2 – Modèle général de la théorie du double codage [adaptée et traduite de Paivio (2007)].

Selon Paivio (2007), les *imagènes* sont des unités de représentation visuelle d'objets qui ont tendance à être perçues de manière imbriquée. Par exemple, il est possible de s'imaginer une personne, la personne dans une voiture et la voiture roulant sur la route. Le traitement des *imagènes* est réalisé de manière holistique, c'est-à-dire qu'ils sont difficilement décomposables en unités discrètes. C'est pourquoi la hiérarchie imposée par l'imbrication d'*imagènes* peut être réalisée par une approche de bas vers le haut, c'est-à-dire de la personne vers la rue, ou du haut vers le bas, c'est-à-dire de la rue vers la personne. Quant aux *logogènes*, il s'agit, toujours selon Paivio (2007), d'unités de représentation verbales telles des lettres, des mots et des phrases. Contrairement aux *imagènes*, ces unités de représentation sont organisées d'une manière séquentielle imposée par des conventions à travers leur hiérarchie d'imbrication. En effet, le sens conventionnel de l'écriture impose une contrainte d'organisation des lettres et des mots, donc des *logogènes*. C'est pour cette raison qu'il est plus facile d'épeler un long mot à l'endroit qu'à l'envers. La distinction entre ces deux formes de représentation interne

d'informations provient du fait qu'elles sont traitées par des régions différentes du cerveau; le traitement des *imagènes* est réalisé dans le cortex visuel, qui est également associé à la mémoire à long terme, et le traitement de *logogènes* est réalisé dans le cortex temporal (Ware 2004, p. 298).

Bien que ces deux types de représentations internes soient associés à différentes régions du cerveau, il existe une relation entre les *imagènes* et les *logogènes*. Une image, par exemple la photographie d'une personne, est un stimulus visuel dont la représentation interne est réalisée par le système non-verbal à l'aide de plusieurs *imagènes* (la personne, ses caractéristiques faciales, le paysage autour de celle-ci, la photographie, la main qui tient la photographie, etc.). Si la personne est connue, son nom peut venir à l'esprit. Toutefois, le nom d'une personne n'est pas une information visuelle, mais verbale. Les *imagènes* associés aux attributs physiques de cette personne sont donc reliés à des *logogènes* associés à, par exemple, son nom, son adresse, son numéro de téléphones, etc. L'inverse est également possible. La représentation mentale du nom d'une personne connue (*logogène*) suite à sa lecture est reliée à des *imagènes* (apparence de cette personne, événements avec cette personne, etc.) associés à cette personne. En somme, les connexions référentielles sont généralement de l'ordre de une à plusieurs ou de plusieurs à plusieurs.

La théorie du double codage permet de comprendre la manière dont les stimuli verbaux et non-verbaux sont traités cognitivement, mais elle ne permet pas de répondre directement à la question posée précédemment. À cette fin, il faut donc se tourner vers la théorie de l'adéquation cognitive[8] proposée par Vessey (1991) qui s'appuie en partie sur la théorie du double codage. Telle qu'illustrée à la figure 2.3, cette théorie indique qu'il existe une corrélation entre la performance de la tâche (solution du problème) et

[8] Traduction libre du terme anglais *cognitive fit*.

l'adéquation (représentation mentale) entre la forme de représentation externe d'un problème (représentation du problème) et la forme de sa représentation requise pour la tâche (tâche de résolution de problème).

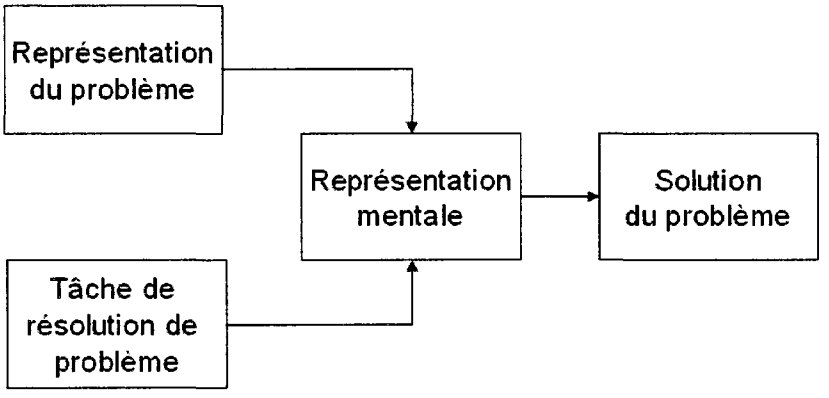


Figure 2.3 – Modèle général de résolution de problème selon la théorie de l'adéquation cognitive [traduite de Vessey (1991)].

Selon la théorie de l'adéquation cognitive, avant de savoir si la forme graphique ou textuelle est préférable, il faut s'interroger sur les caractéristiques de la tâche dans laquelle le modèle sera utilisé. À ce sujet, Ware (2004, p. 304) indique qu'il est préférable de recourir à la forme graphique pour représenter des relations structurelles, des emplacements et des détails. Ces représentations, de nature spatiale, requièrent une approche holistique de traitement. Quant à la forme textuelle, il indique qu'elle est préférable pour représenter des informations procédurales, des conditions logiques et des concepts verbaux abstraits. Ces représentations, de nature symbolique, requièrent une approche séquentielle de traitement. En ce qui concerne particulièrement les instructions procédurales, Ware (2004, p. 305) indique qu'il est possible d'utiliser la forme graphique à condition qu'elle soit dynamique.

Tel qu'illustré à la figure 2.2, la théorie du double codage postule que les stimuli verbaux et non-verbaux sont traités cognitivement par des canaux différents qui se

partagent les ressources attentionnelles. Par exemple, il est difficile de mémoriser l'information perçue visuellement en regardant une émission de télévision présentant des images de pyramides égyptiennes et dont le narrateur raconte la vie d'un acteur hollywoodien. Dans ce cas, les *logogènes* obtenus suite à l'écoute du narrateur ne correspondent pas aux *imagènes* obtenus suite aux images présentées et la rétention sera médiocre. Par contre, si le narrateur porte son discours sur les pyramides égyptiennes, la rétention sera de meilleure qualité que si un seul type de stimuli avait été perçu. En somme, la rétention d'information est de meilleure qualité lorsque la même information est présentée de manière verbale et non-verbale.

2.1.2. Un modèle est une simplification

Un concept d'importance capitale en génie logiciel est la séparation des préoccupations. Une préoccupation peut être considérée comme un ensemble de caractéristiques d'un logiciel pouvant évoluer indépendamment d'un autre ensemble de caractéristiques, c'est-à-dire d'une autre préoccupation. Par exemple, Kulkarni et Reddy (2003) présentent l'IU, les traitements et les données comme étant trois préoccupations. Particulièrement reliées aux IU, la littérature présente différentes préoccupations pouvant être modélisées telles les contextes d'usage (Sottet, Clavary et Favre 2005), la tâche (Pinheiro da Silva et Patton 2003), les interactions (Nunes et Cuhna 2000) ainsi que la navigation et la présentation (Hennicker et Koch 2001). En somme, une préoccupation fait référence à la manière de voir le système informatique, c'est-à-dire la perspective adoptée. La séparation des préoccupations vise donc à regrouper les éléments d'un logiciel autour de différentes préoccupations afin de pouvoir optimiser leur fonctionnement sans avoir à se préoccuper des autres éléments associés à des préoccupations différentes.

En faisant abstraction des éléments qui ne sont pas rattachés à la perspective adoptée, la représentation d'un objet est donc simplifiée. Ainsi, selon Kühne (2006), une copie d'un

objet ne peut être considérée comme étant un modèle; la copie n'est pas une représentation, mais une imitation de l'objet.

La figure 2.4 illustre la notion de simplification. L'ellipse grise dans l'objet est la partie de celui-ci qui concerne la perspective adoptée. Le modèle est représenté par une ellipse plus petite représentant la simplification de l'objet modélisé. Les trois flèches représentent les activités de dénotation, de démonstration et d'interprétation telles qu'illustrées à la figure 2.1.

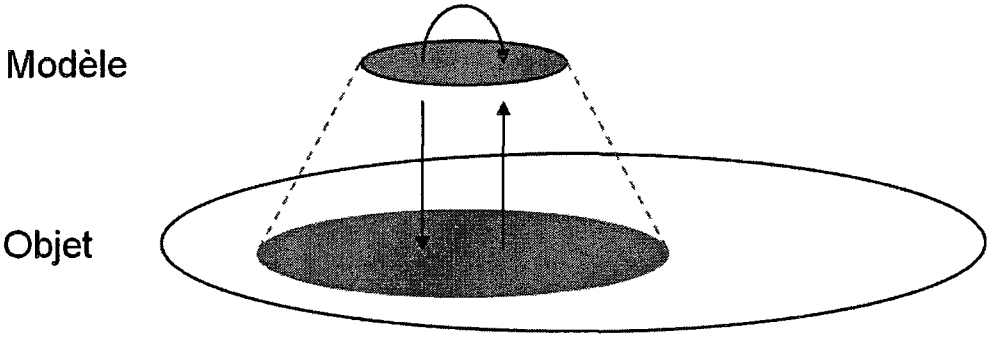


Figure 2.4 – Simplification d'un objet.

La simplification de l'objet modélisé a pour effet d'offrir, entre autres, un mécanisme de gestion de la complexité. À ce sujet, Bézivin et Gerbé (2001) indiquent que pour être utile, un modèle doit être plus facile à utiliser que l'objet lui-même. Par exemple, le concepteur de base de données ne s'intéressera pas aux aspects reliés à l'IU d'un logiciel, mais seulement aux aspects qui portent sur le stockage et l'accès aux données. Il utilisera donc un modèle qui se rapporte à cette perspective uniquement. En somme, un modèle est un moyen économique d'étudier un objet; il permet de substituer celui-ci dans son étude.

2.2. Qu'est-ce qu'un langage de modélisation?

Mellor, Scott, Uhl et Weise (2004) distinguent trois types de modèles en génie logiciel : croquis, plan et exécutable. Un modèle en tant que croquis est réalisé rapidement; il ne s'agit pas d'une représentation précise ni complète. Un modèle en tant que plan permet de construire quelque chose, en l'occurrence un logiciel. Celui-ci comporte donc un niveau de détail et de rigueur beaucoup plus élevé qu'un croquis. Un modèle en tant qu'exécutable peut être directement traduit en code source. Pour ce faire, il exige d'être réalisé sans ambigüité de manière à pouvoir être traité par un logiciel de transformation à cette fin.

Un modèle peut être utilisé par différents individus à des fins d'étude de l'objet, pas uniquement par celui qui l'a créé; dans ce cas, un bon modèle devient un outil de communication (Mellor, Scott, Uhl et Weise 2004). Par conséquent, quel que soit le type de modèle, afin d'être interprété correctement, il est nécessaire que le langage de modélisation utilisé soit compris sans ambigüité. Pour des modèles de type croquis, les langages de modélisation se limitent à quelques conventions, mais ne sont pas définis formellement. Toutefois, pour des modèles de type plan, mais surtout de type exécutable, ceux-ci doivent s'appuyer sur une notation et des règles de modélisation strictes. Cette section porte sur les langages de modélisation. Plus spécifiquement, elle porte sur la manière dont ces langages sont définis.

2.2.1. Modèles et méta-modèles

Afin de pouvoir les interpréter correctement, les modèles s'appuient sur un ensemble de règles préalablement défini qui impose une structure et une sémantique. Un méta-modèle permet d'atteindre cet objectif en spécifiant les concepts du langage employés pour modéliser les relations entre les instances de ceux-ci (Seidewitz 2003, Mellor,

Scott, Uhl et Weise 2004). En d'autres mots, un méta-modèle est un modèle du langage de modélisation.

Bézivin et Gerbé (2001) présentent la relation entre objet, modèle et méta-modèle. La figure 2.5, une variante de la figure 2.1, illustre cette relation. Le méta-modèle qualifie la relation entre le modèle et l'objet modélisé.

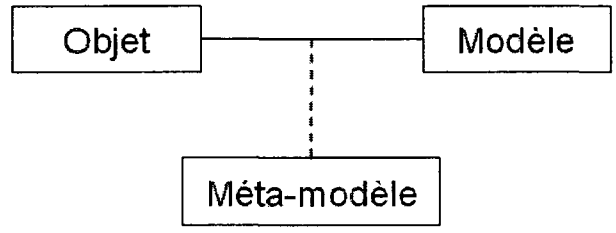


Figure 2.5 – Relation entre objet, modèle et méta-modèle [adaptée et traduite de Bézivin et Gerbé (2001)].

La définition d'un langage de modélisation ne se limite pas simplement à définir un ensemble de symboles. Il faut également définir des règles d'agencement entre ceux-ci d'une manière permettant d'être interprété correctement ainsi qu'une notation permettant d'être utilisé. À cette fin, la syntaxe d'un langage de modélisation comporte deux niveaux (Kelly et Tolvanen 2008, Mellor, Scott, Uhl te Wise 2004, Frankel 2003) :

- **Syntaxe abstraite.** Ce type de syntaxe définit les éléments du langage de modélisation et les relations entre ceux-ci. Elle peut s'apparenter au vocabulaire et à la grammaire d'un langage. Un méta-modèle sert à définir la syntaxe abstraite d'un langage de modélisation.
- **Syntaxe concrète.** Ce type de syntaxe définit l'apparence des éléments de la syntaxe abstraite. Elle consiste en la définition d'une notation symbolique pouvant être manipulée pour réaliser des modèles et être interprétée lors de l'étude de ces derniers.

La figure 2.6 illustre un langage de modélisation pour l'assignation de tâches à des employés. La syntaxe abstraite, dont le méta-modèle est représenté par un diagramme de classes du langage UML, présente les éléments du langage à représenter et la manière dont ceux-ci peuvent être reliés. Par exemple, un employé peut être relié à aucune ou plusieurs tâches et une tâche peut être reliée à un ou plusieurs employés. La syntaxe concrète est définie par des bonhommes fil de fer pour les employés, des rectangles pour les tâches et des lignes pour les assignations.

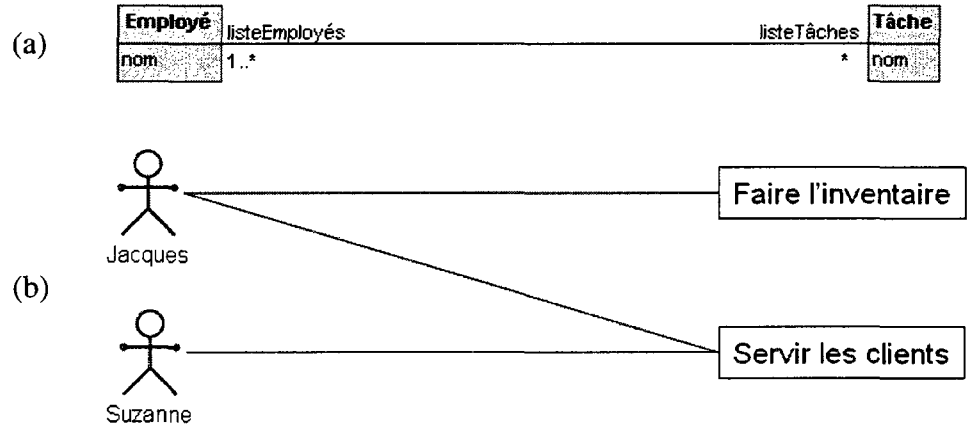


Figure 2.6 – Langage de modélisation pour l'assignation de tâches : (a) la syntaxe abstraite et (b) la syntaxe concrète.

Un méta-modèle sert non seulement à définir un langage de modélisation d'une manière formelle, mais permet également de le faire évoluer. Par exemple, si un nouveau concept doit être représenté par le langage de modélisation, un nouvel élément représentant le concept est ajouté au méta-modèle, permettant ainsi de définir comment ce nouveau concept est relié aux concepts déjà existants. Par exemple, la figure 2.7 illustre l'ajout d'employés à temps plein et d'employés à temps partiel au méta-modèle. La syntaxe concrète permet de distinguer ces deux éléments par un bonhomme fil de fer complet

pour l'employé à temps plein et un demi bonhomme fil de fer pour l'employé à temps partiel. Ainsi, on remarque que Jacques est un employé à temps plein et Suzanne une employée à temps partiel.

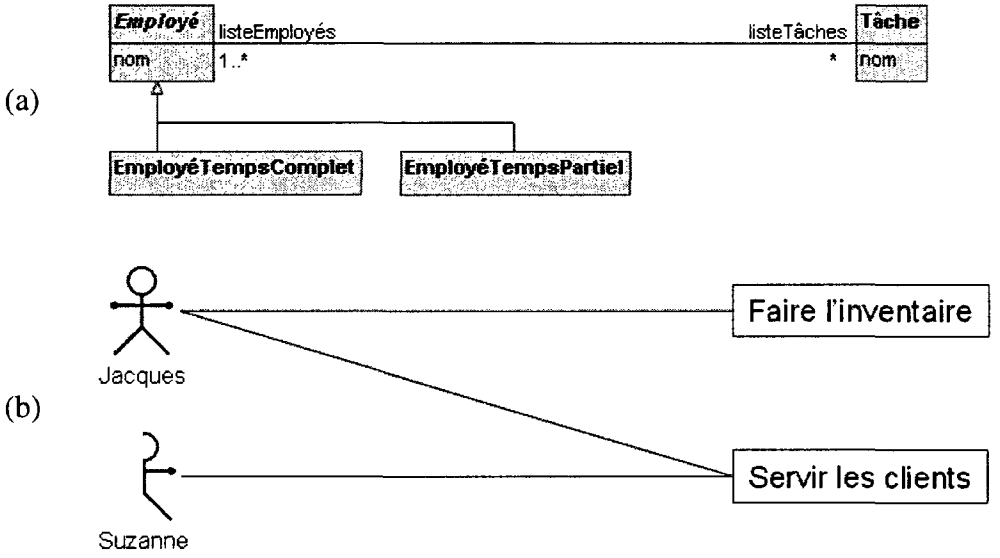


Figure 2.7 – Nouveau langage de modélisation pour l'assignation de tâches : (a) la syntaxe abstraite et (b) la syntaxe concrète.

2.2.2. Les niveaux de modélisation

Un méta-modèle étant lui-même un modèle peut être méta-modélisé afin d'en définir la structure et la sémantique. Où s'arrête la méta-modélisation? L'Object Management Group (OMG) (2008a) définit une architecture à quatre niveaux de modélisation présentée au tableau 2.1. Il est à noter que ces niveaux ont été définis spécifiquement pour le domaine du génie logiciel.

Tableau 2.1 – Les quatre niveaux de modélisation selon OMG (2008a).

Niveaux	Appelations	
M0	Données	-
M1	Métadonnées	Modèle
M2	Méta-métadonnées	Méta-modèle
M3	Méta-méta-métadonnées	Méta-méta-modèle

- **M0.** Ce niveau décrit les données du logiciel lors de son exécution. Celles-ci représentent des objets du domaine d'application. Elles peuvent être représentées au niveau M1.
- **M1.** Ce niveau comporte les modèles qui représentent les données du niveau M0. En d'autres mots, le niveau M1 représente les métadonnées, c'est-à-dire les données décrivant les données du niveau M0. Ces modèles présentent une vue abstraite de la structure et du comportement du logiciel.
- **M2.** Ce niveau sert à décrire les langages utilisés au niveau M1. La structure et la sémantique des modèles du niveau M1 y sont décrites. Les modèles à ce niveau sont considérés comme constituants un langage abstrait, c'est-à-dire un langage sans définition de syntaxe concrète.
- **M3.** Ce niveau définit un langage permettant de définir les modèles au niveau M2. Comme le niveau M3 est décrit à partir de lui-même, il n'est pas nécessaire de recourir à des niveaux supérieurs de modélisation.

Plusieurs avantages sont associés à cette architecture à quatre niveaux (Frankel 2003, Mellor, Scott, Uhl et Weise 2004, OMG 2008a). Premièrement, elle permet de soutenir n'importe quelle sorte de modèles et de paradigmes de modélisation inimaginable. Deuxièmement, elle permet à plusieurs sortes de métadonnées d'être reliées. Troisièmement, elle permet d'ajouter des méta-modèles et de nouvelles sortes de

métadonnées de manière incrémentale. Quatrièmement, elle permet l'échange de modèles et de méta-modèles entre entités utilisant le même méta-méta-modèle.

2.2.3. Langages de modélisation génériques et langages de modélisation spécifiques au domaine

Il existe deux types de langages de modélisation : les langages de modélisation génériques (LMG) et les langages de modélisation spécifiques au domaine (LMSD). Les LMG sont des langages qui peuvent être utilisés dans différents domaines. Par exemple, le diagramme de classes du langage UML (OMG 2008b) ou le diagramme entité-association (Chen 1976) sert à représenter la structure conceptuelle d'un domaine quel qu'il soit, par exemple un système de création d'horaire académique, un système de gestion de comptes bancaires ou un système de réservation hôtelier. De plus, ces diagrammes peuvent servir à représenter différents aspects de l'architecture comme par exemple les objets internes utilisés par le logiciel dans le cas du diagramme de classes. L'avantage de ce type de langage de modélisation est que le modélisateur n'a qu'un seul langage à connaître pour modéliser différentes situations.

L'inconvénient des LMG est que, dans certains cas, ceux-ci sont limités quant à leur capacité de représenter tous les éléments d'un domaine particulier. En effet, puisque ce type de langage consiste en une abstraction des concepts d'implémentation, il oblige le modélisateur à traduire sa compréhension du domaine en des concepts qui sont associés à leur implémentation (Kelly et Tolvanen 2008, p. 55). Par exemple, les éléments du diagramme de classes (classe, attribut, opération, généralisation, ...) utilisés pour représenter les concepts d'un domaine se rapportent à des éléments qui se retrouvent dans l'implémentation sous forme de code source. Même son de cloche pour le diagramme entité-association et l'implémentation de la base de données.

Quant aux LMSD, ils représentent directement les concepts associés au domaine visé sans égard à leur implémentation (Kelly et Tolvanen 2008, p. 15). Ainsi, les développeurs prennent le rôle de modélisateur pour créer des logiciels en manipulant directement des concepts du domaine. Par exemple, pour modéliser le fonctionnement d'une montre à cristaux liquides, le modélisateur manipule des concepts tels des boutons, des icônes d'affichage, des états et des actions de l'utilisateur (Kelly et Tolvanen 2008, ch. 9). Le langage de modélisation pour l'assignation des tâches illustré aux figures 3.7 et 3.8 est un LMSD. Afin de tirer les avantages d'un LMSD, celui-ci est nécessairement défini par un méta-modèle (Kelly et Tolvanen 2008, p. 74, Schmidt 2006).

Tout comme pour les LMG, la notion de domaine est très large. Un LMSD peut servir à représenter les éléments d'un domaine d'application comme un système de services financiers en ligne ou les éléments d'un domaine d'implémentation comme des composants logiciels (Balasubramanian, Gokhale, Karsai, Sztipanovits et Neema 2006, Schmidt 2006).

2.3. Différents modèles pour différentes étapes

Tout processus de création d'artéfacts consiste en la transformation d'un espace de problème en un espace de solution[9] et la concrétisation de celle-ci. La création de la représentation de l'espace de problème est appelé l'analyse et la création de la représentation de l'espace de solution est appelée la conception. Chacune de ces deux étapes consiste à représenter une perspective particulière.

[9] Généralement, le terme *espace de solution* est employé au pluriel pour faire référence aux différentes solutions pouvant répondre au problème. Toutefois, comme le processus de création d'artéfact se limite généralement à une seule solution, le terme est employé au singulier.

2.3.1. Modèles d'analyse

L'espace de problème est obtenu suite à une phase d'analyse qui, selon Odell et Ramackers (1997), consiste en la transformation d'une perception du monde réel en une représentation de celui-ci. En d'autres mots, il s'agit du « quoi? ». Par conséquent, de par sa nature, un modèle est subjectif; il est le produit de la compréhension du monde réel par son auteur. Comment dire qu'un modèle est complet? La modélisation est un processus itératif. À chaque itération, une validation doit être réalisée, généralement par des experts du domaine d'application. La validation fait partie de l'activité de démonstration telle qu'illustrée à la figure 2.1. C'est lorsque les experts du domaine ont donné leur aval que le modèle peut être considéré comme complet et sans erreurs. L'est-il vraiment? Rien ne peut garantir la qualité d'un modèle évalué qualitativement. Selon Reason (1995), il faut toujours garder à l'esprit qu'il existe une probabilité non nulle qu'un individu fasse une erreur lors de l'exécution de ses tâches. Il importe donc de l'outiller de manière à ce que celui-ci puisse rapidement corriger le tir. C'est pourquoi il est nécessaire de permettre le retour en arrière afin de modifier le modèle à tout moment.

2.3.2. Modèles de conception

L'espace de solution est obtenu suite à une phase de conception qui, toujours selon Odell et Ramackers (1997), consiste en la transformation du résultat de l'analyse en une représentation de l'artéfact, plus spécifiquement une représentation de la structure de composants qui, une fois réunis, permettent de résoudre le problème. En d'autres mots, il s'agit du « comment? ». Ainsi, les modèles de conception sont tributaires des modèles d'analyse. Par conséquent, pour juger de la qualité d'un modèle de conception, il ne faut pas évaluer si l'artéfact conçu permet de solutionner le problème, mais plutôt s'il permet de solutionner le problème tel que décrit par le modèle d'analyse. En d'autres mots, si le modèle d'analyse est erroné, le modèle de conception ne pourra pas permettre de représenter l'artéfact qui solutionnera le problème.

2.4. Ingénierie dirigée par les modèles

L'ingénierie dirigée par les modèles (IDM) est une nouvelle approche de développement de logiciels qui consiste à mettre l'accent, comme son nom l'indique, sur les modèles plutôt que sur le code source. En résumé, l'IDM consiste à définir un ensemble de modèles successifs et un ensemble de règles de transformation permettant d'arriver à des modèles cibles à partir de modèles sources (Kent 2002, Mellor, Clark et Futagami 2003, Schmidt 2006). Il s'agit donc de modèles exécutables.

Afin de tirer profit de l'IDM, il est essentiel d'avoir des outils automatisant les transformations entre modèles pour ultimement arriver à l'artéfact final, c'est-à-dire le code source[10] dans le cas d'un logiciel. Ces outils, en appliquant les règles de transformation, sont responsables de préserver la cohérence entre les différents modèles.

2.4.1. L'évolution du développement de logiciels

Pour comprendre d'où provient l'IDM, Frankel (2003) ainsi que Mellor, Scott, Uhl et Weise (2004) présentent l'évolution du développement de logiciels telle qu'illustrée à la figure 2.8. Cette évolution porte sur l'augmentation graduelle du niveau d'abstraction requis pour représenter un logiciel, autant sa structure que son comportement.

Les premiers ordinateurs étaient programmés à partir d'un langage machine, appelées langages de première génération (L1G), consistant à aligner une série de 0 et de 1 formant des instructions binaires. Ces programmes étaient réalisés à partir de connections entre fils électriques formant un ensemble de commutateurs. Le niveau d'abstraction était donc limité au matériel.

[10] Le code source, habituellement écrit avec un langage de troisième génération, est une représentation de l'exécution du logiciel. Il est donc considéré comme un modèle de l'exécution de celui-ci.

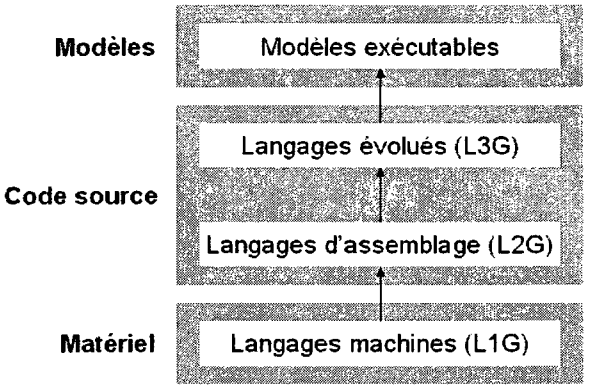


Figure 2.8 – Évolution des langages de représentation du logiciel.

Dans le but de réduire la complexité associée à l'élaboration d'instructions en langage machine, la deuxième génération de langages de programmation (L2G) a fait son apparition. Les langages d'assemblage permettent de générer du langage machine à partir d'instructions simples pouvant être comprises par un individu. De plus, il est possible, contrairement au L1G, d'utiliser le même code en langages d'assemblage sur différentes versions du processeur; les instructions binaires sont réorganisées de manière à respecter les instructions écrites en langage d'assemblage. Cette catégorie de langages a ouvert la porte à un nouveau niveau d'abstraction : le code source.

Les L2G devenaient insuffisants pour deux principales raisons. Premièrement, puisque ceux-ci sont une simplification des L1G, ils sont limités à un type spécifique de plateforme de matériel. Deuxièmement, la complexité grandissante des logiciels faisait en sorte qu'il devenait laborieux de les écrire en un L2G. Il fallait donc, encore un fois, augmenter le niveau d'abstraction. Toujours selon le même paradigme, les langages évolués, tels FROTRAN, COBOL, C/C++ et Java pour n'en nommer quelques-uns, ont fait leur apparition. Ces langages de troisième génération (L3G) proposent une syntaxe se rapprochant de la manière dont peut être formulée une solution au problème qui

nécessite l'écriture du programme. Comme ces langages sont plus près du domaine d'application que de la plateforme, il est nécessaire de réaliser des correspondances avec le langage machine. À cette fin, un ensemble d'outils ont fait leur apparition pour soutenir les L3G, notamment des assembleurs, préprocesseurs et compilateurs. Ces outils permettent entre autres de transformer une instruction écrite avec un L3G en un ensemble d'instructions en L2G. À leur tour, chaque instruction en L2G est transformée en un ensemble d'instructions en L1G afin que l'ordinateur puisse exécuter le programme en question.

Les L3G ont permis d'introduire des concepts comme la réutilisation et l'interopérabilité. La réutilisation consiste à pouvoir reprendre une section d'un programme et l'utiliser dans un autre programme. Le paradigme objet est propice à la réutilisation car il permet de regrouper des éléments de programmes sous la forme d'objets. Ces derniers représentent des éléments d'un logiciel ou du domaine d'application et peuvent être utilisés à nouveau dans d'autres programmes. Quant à l'interopérabilité, il s'agit de la possibilité d'exécuter le même programme sur différentes plateformes. Par exemple, grâce à sa machine virtuelle, le langage Java permet l'exécution du même code source sur pratiquement n'importe quel système d'exploitation.

Aujourd'hui, les logiciels peuvent s'appuyer sur une architecture distribuée, contenir plusieurs centaines de milliers de lignes de code, inclure un grand nombre de fonctionnalités qui s'adressent à plusieurs types d'utilisateurs à la fois et être exécutés sur différentes plateformes. Afin de gérer la complexité inhérente au développement de logiciels, les concepteurs s'orientent vers la modélisation. Les modèles permettent une vue simplifiée du logiciel à concevoir afin de permettre aux concepteurs de concentrer leurs efforts sur des aspects particuliers de celui-ci. Toutefois, dans une approche

orientée vers le code source, les modèles ne sont utilisés que pour soutenir le développement; les modèles sont rarement liés de manière explicite avec les L3G de la même manière que ces derniers le sont avec les L2G qui, eux-mêmes, le sont avec les L1G. L'IDM consiste donc à augmenter le niveau d'abstraction une fois de plus pour mettre l'accent cette fois-ci sur les modèles et non le code source. D'après cette approche, le développement se réalise en modélisant le logiciel. À partir de règles de transformations spécifiques, ces modèles, dont les langages respectifs ont été rigoureusement définis sous forme d'un méta-modèle, peuvent être transformés en L3G pour ensuite être transformés en L2G pour, dernièrement, être transformés en L1G afin d'être exécutés. En somme, les modèles deviennent exécutables.

2.4.2. Les avantages

La littérature rapporte plusieurs avantages à cette approche. Cette section en présente quelques-uns :

- **Synchronisation entre les modèles et leur implémentation.** Kent (2002) indique la synchronisation entre les modèles et leur implémentation comme avantage. Dans une approche de développement traditionnelle, les modèles sont réalisés en marge de leur implémentation et leur synchronisation doit être réalisée manuellement, ce qui est très laborieux et donc souvent escamoté. Ceci résulte en des modèles qui ne correspondent plus à leur implémentation après un certain temps puisque certaines modifications réalisées à même le code source ne sont souvent pas accompagnées d'une mise à jour dans les modèles.
- **Meilleure réutilisation d'artéfacts.** Mellor, Scott, Uhl et Weise (2004) parlent d'une meilleure réutilisation des artéfacts. En effet, la synchronisation entre les modèles et leur implémentation fait en sorte que chaque élément d'un modèle est associé à des représentations à des niveaux d'abstraction inférieurs. Par exemple, le

symbole d'un bouton ou d'une liste déroulante est associé à des composants logiciels sous forme de code source. Ainsi, ces éléments d'IU peuvent être réutilisés aisément. Il en va de même pour des composants se rapportant au domaine d'application tel un client ou un compte qui s'accompagne de composants logiciels.

- **Artéfacts de qualité.** Selon Selic (2003), le code source généré est, en principe, sans erreur puisqu'il a été utilisé à plusieurs reprises. Toutefois, ceci ne veut pas dire qu'un logiciel développé à partir de cette approche est sans erreurs, mais la correction de celles-ci se rapporte à l'implémentation des concepts du domaine d'application plutôt qu'aux éléments de l'architecture logicielle en place.
- **Manipulation des concepts du domaine d'application.** Tous ces avantages permettent donc de manipuler des concepts du domaine d'application plutôt que des concepts liés à leur implémentation qui, selon Selic (2003), est le principal avantage d'une telle approche de développement.
- **Réduction du cycle de développement.** Selic (2003) indique également qu'un avantage des modèles exécutables est la possibilité de voir le produit final beaucoup plus rapidement. En effet, le code source exécutable est obtenu d'une manière automatisée plutôt que d'être écrit par un programmeur. Le cycle de développement en est donc énormément raccourci et une validation du produit complet peut être réalisée plus rapidement.

En somme, tous ces avantages contribuent à un accroissement de la productivité du développement de logiciels.

2.4.3. Modèles graphiques ou textuels?

Cette approche de développement de logiciels met en relation les deux formes de modèles : graphique et textuelle. Un modèle créé par un individu doit pouvoir être traité par un logiciel de transformation. Tel que mentionné précédemment, la perception chez

un individu consiste en un traitement parallèle. Ainsi, il est possible pour un individu de traiter un modèle graphique. Toutefois, un ordinateur n'est capable que de traitement sériel. Par conséquent, un même modèle doit pouvoir être présenté autant graphiquement que textuellement. Cette conversion est en fait associée à la syntaxe concrète; il s'agit de modifier la notation utilisée pour représenter les éléments du modèle. En somme, plusieurs syntaxes concrètes, par exemple une graphique et une textuelle, peuvent être associées à une même syntaxe abstraite. La figure 2.9 illustre la représentation textuelle du modèle de la figure 2.6. Chaque élément de la représentation textuelle correspond à un élément du méta-modèle[11].

2.4.4. Perspectives pour l'avenir

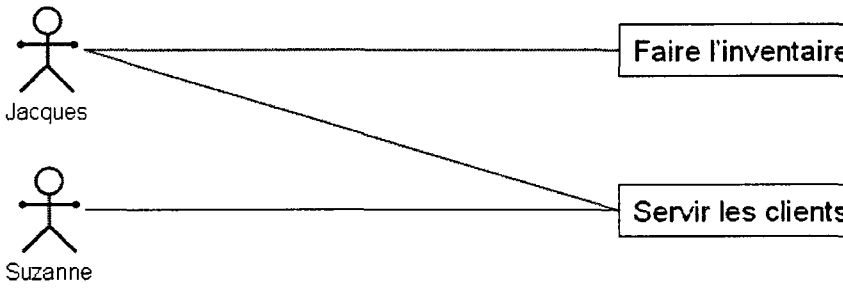
Est-ce que l'IDM n'est qu'un vœu pieux? Cette question n'est pas sans fondements puisque plusieurs initiatives aux allures prometteuses en génie logiciel ont rapidement sombré dans l'oubli. Néanmoins, plusieurs points semblent indiquer que dans un avenir rapproché le développement de logiciels réalisé à partir d'une approche d'IDM sera courant. Premièrement, tel que décrit précédemment, l'histoire du développement de logiciel indique que cette approche est une suite logique dans l'élévation du niveau d'abstraction. Deuxièmement, la prolifération d'ouvrages tout aussi bien professionnels qu'académiques sur le sujet ainsi que le nombre d'outils sur le marché s'affichant comme soutenant cette approche indiquent un intérêt marqué pour l'IDM. Troisièmement, des sociétés de veille technologique telles Gartner (Fenn 2007) et Forrester (Lo Giudice 2007) voient l'IDM comme une technologie émergente et prévoient son utilisation courante d'ici les prochaines années.

---

[11] Il est à noter que les identifiants (« id ») utilisés dans la représentation textuelle ne correspondent à aucun élément du méta-modèle. Normalement, chacune des trois classes du méta-modèle devrait avoir un attribut « id ». Toutefois, celui-ci a été omis pour des raisons de simplicité. En effet, l'identifiant n'est pas utilisé pour la représentation du domaine; il sert uniquement à des fins de relation entre les éléments pour la représentation textuelle.

(a)

(b)



(c)

```
<Employé id="e1" nom="Jacques">
   <listeTâches>
      <Tâche id="t1" />
      <Tâche id="t2" />
   </listeTâches>
</Employé>
<Employé id="e2" nom="Suzanne">
   <listeTâches>
      <Tâches id="t2" />
   </listeTâches>
</Employé>
<Tâche id="t1" nom="Faire l'inventaire">
   <listeEmployés>
      <Employé id="e1" />
   </listeEmployés>
</Tâche>
<Tâche id="t2" nom="Servir les clients">
   < listeEmployés >
      < Employé id="e1" />
      < Employé id="e2" />
   </listeEmployés >
</Tâche>
```

Figure 2.9 –Langage de modélisation pour l'assignation de tâches : (a) la syntaxe abstraite, (b) la syntaxe concrète graphique et (c) la syntaxe concrète textuelle.

2.5. Synthèse

Un modèle est une représentation simplifiée d'un objet. Il s'agit d'un moyen économique d'étudier ce dernier. Un modèle peut être graphique ou textuel. La forme

graphique est préférable pour des représentations structurelles tandis que la forme textuelle est préférable pour des représentations procédurales.

Un modèle est également utilisé comme outil de communication. À cette fin, celui-ci doit s'appuyer sur un langage de modélisation dont la compréhension est partagée. Pour ce faire, la syntaxe abstraite du langage de modélisation doit être rigoureusement définie. Un méta-modèle, c'est-à-dire un modèle du langage de modélisation, est habituellement utilisé à cette fin. Il s'agit d'une définition formelle du langage de modélisation. Afin de pouvoir utiliser le langage, une syntaxe concrète, c'est-à-dire une notation symbolique, doit être associée à la syntaxe abstraite. Un méta-modèle est, de par sa nature, évolutif; il est possible de modifier le langage de modélisation en modifiant le méta-modèle.

L'intérêt de définir rigoureusement un langage de modélisation est entre autres justifié par une nouvelle approche de développement, appelée IDM, qui met les modèles au premier plan. En toute vraisemblance, cette approche sera courante dans un avenir rapproché. Elle consiste à définir un ensemble de modèles successifs et un ensemble de règles de transformation permettant d'arriver à des modèles cibles à partir de modèles sources. Afin d'automatiser ces transformations, les langages de modélisation utilisés doivent s'appuyer sur une syntaxe abstraite rigoureuse. La méta-modélisation est donc indispensable à cette fin.

CHAPITRE 3 - LES INTERFACES ÉCOLOGIQUES

Afin de soutenir un individu en situation de résolution de problème, une IU doit représenter la structure du domaine de travail. Les IE correspondent à ce type d'IU. Elles permettent à leurs utilisateurs non seulement de faire face à des situations prévues, mais surtout de s'adapter lors de situations imprévues.

Ce chapitre présente une revue de la littérature sur les IE en s'intéressant particulièrement à leur structure. Afin de bien saisir la contribution d'une IE à la résolution de problème, il importe de définir ce processus cognitif en premier lieu. La section 3.1 porte sur ce sujet en s'appuyant sur la théorie de Newell et Simon (1972). Cette description ne se veut pas exhaustive, mais juste assez élaborée pour comprendre cette théorie ainsi que certaines notions complémentaires. L'objectif n'est pas de présenter une critique de la théorie de Newell et Simon (1972), mais plutôt d'aider le lecteur à comprendre les concepts ultérieurement présentés dans cette thèse. La section 3.2 présente une description des IE. À partir de cette dernière, un méta-modèle des IE est présenté à la section 3.3. Ce méta-modèle consiste en une intégration des notions associées aux IE qui sont présentées dans la littérature. Il s'agit d'une contribution de cette thèse.

3.1. La résolution de problème

Anciennement utilisé pour représenter le processus décisionnel d'un individu, le modèle classique du décideur rationnel (*homo economicus*), de nature normative et théorique, a été remplacé par un modèle descriptif et, par le fait même, plus réaliste qui s'appuie sur le concept de rationalité limitée (March et Simon 1958, Simon 1997). Ce modèle reconnaît que les individus sont victimes de contraintes cognitives (mémoire, attention,

perception, etc.) et environnementales (temps, accès à l'information, etc.) qui les empêchent par eux-mêmes d'arriver à des décisions qualifiées d'optimales. C'est pourquoi ils font plutôt appel à une stratégie décisionnelle visant à satisfaire un ensemble de critères (Simon 1956, March et Simon 1958, p. 140). Cette représentation du décideur est à la base de la théorie de résolution de problèmes par l'humain de Newell et Simon (1972).

3.1.1. Système de traitement d'informations

La théorie de résolution de problèmes par l'humain de Newell et Simon (1972) postule que l'individu est un système de traitement d'informations (STI). Un STI, tel qu'illustré à la figure 3.1, est en mesure de traiter des symboles. Ces derniers peuvent être divisés en deux catégories : les jetons et les types. Un jeton est un élément pouvant être comparé par le STI afin de dire s'il est égal ou différent. Un ensemble de jetons déterminés comme étant identiques est appelé un type. Une structure de symboles consiste en des jetons et des relations. Newell et Simon (1972, p. 22) donne comme exemple la lettre « S » qui se traduit par « — — — » en code Morse, une structure de symboles de trois jetons du même type associés par la relation *suivant*.

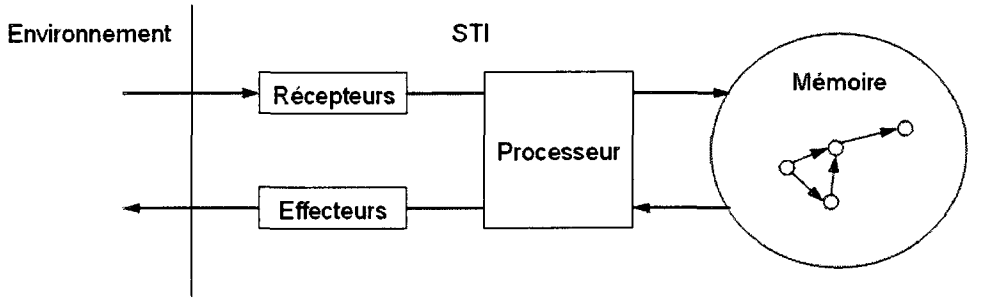


Figure 3.1 – Système de traitement d'informations [traduit de Newell et Simon (1972)].

Les structures de symboles sont associées les unes aux autres par désignation, c'est-à-dire une relation de référence. Newell et Simon (1972, p. 24) donnent comme exemple la

structure de symboles « couleur, maison » pouvant être associée au symbole « blanc ». En d'autres mots, la structure de symboles « couleur, maison » *désigne* le symbole « blanc » qui, inversement, *désigne* la couleur de la maison.

Les structures de symboles peuvent désigner des configurations sensorielles familières comme des lettres ou des objets, mais également des instructions permettant de transformer des structures de symboles en de nouvelles. Ces instructions de transformation, faisant référence à des traitements, sont désignées par des structures de symboles appelées des programmes dans le jargon de Newell et Simon (1972, p. 21).

Afin de traiter les structures de symboles, un STI s'appuie sur un certain nombre de composants identifiés à la figure 3.1 :

- **Récepteurs.** Les récepteurs perçoivent l'état de l'environnement sous forme de symboles désignés en mémoire à long terme.
- **Processeur.** Le processeur est le centre de traitement des symboles. Les symboles utilisés par les processus ou résultant de ceux-ci sont contenus dans une mémoire à court terme faisant partie du processeur. La mémoire à court terme ne peut contenir qu'un petit nombre de structures de symboles à la fois, sept plus ou moins deux selon Miller (1956). Le traitement est réalisé de manière sérielle; une seule opération élémentaire peut être réalisée à la fois. Par exemple, Newell et Simon (1972, p. 796) indiquent que le temps nécessaire pour déterminer lesquels d'un ensemble de n nombres sont divisibles par 7 est linéairement proportionnel à n. En d'autres mots, chaque nombre est évalué un à la suite de l'autre. Dans certains cas, les traitements peuvent utiliser des symboles provenant directement de la mémoire à long terme. Par exemple, Newell et Simon (1972, p. 798) indiquent qu'une personne connaissant la table de multiplication par 12 n'aura pas à calculer, mais plutôt à chercher le résultat.

- **Mémoire.** La mémoire est responsable du stockage à long terme des structures de symboles, programmes ou non. Une quantité illimitée de symboles peut y être stockée.
- **Effecteurs.** Les effecteurs agissent sur l'environnement. Ils permettent d'en modifier l'état.

En plus de ces composants, Newell et Simon (1972) reconnaissent l'importance de la mémoire externe comme par exemple du papier ou un échiquier. La mémoire externe permet de compenser pour la lenteur de l'écriture à la mémoire à long terme et la petite taille de la mémoire à court terme. Elle est ainsi vue comme étant une extension à la mémoire à court terme limitée par le champ de vision de l'individu.

3.1.2. Qu'est-ce qu'un problème?

La figure 3.2 illustre d'une manière simplifiée le processus de résolution de problèmes. Selon Newell et Simon (1972, p. 72), un problème consiste en un écart entre un état actuel de l'environnement (état initial) et son état désiré, souvent appelé le but à atteindre, et pour lequel aucune série d'actions n'est connue pour y arriver. Les actions sont réalisées directement sur l'environnement et permettent d'en modifier son état.

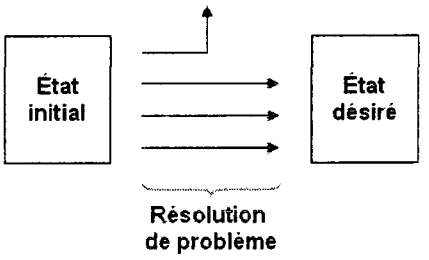


Figure 3.2 – Représentation simplifiée du processus de résolution de problème.

Généralement, il existe plusieurs possibilités permettant d'atteindre le but. Celles-ci sont représentées à la figure 3.2 par des flèches; chacune d'entre elles représente une série d'actions. La résolution de problèmes consiste à définir ces différentes possibilités. Quant à la décision, elle consiste à choisir une de ces possibilités d'actions. Normalement, la possibilité représentée par la première flèche sera exclue car elle ne permet pas d'atteindre le but.

À partir de cette description simplifiée du processus de résolution de problèmes, deux questions peuvent se poser. La première porte sur la manière dont un individu arrive à définir un ensemble de possibilités pour arriver à l'état désiré. La deuxième porte sur la manière dont un individu choisit une solution parmi cet ensemble de possibilités. Le contenu des sections suivantes permet de répondre à ces deux questions.

Avant de procéder, il est essentiel de distinguer les problèmes bien structurés des problèmes mal structurés; chaque type de problème comporte des particularités dans son traitement. Selon Newell et Simon (1972, p. 73), un problème bien structuré en est un pour lequel il existe un test, réalisable sans trop d'effort par le STI, permettant de savoir si une structure de symboles proposée est une solution. Dorst (2006), en citant Simon (1973), ajoute qu'un problème bien structuré, entre autres, n'implique aucun apprentissage ni de redéfinition du problème pendant sa résolution et que toutes les informations nécessaires sont disponibles au préalable. De plus, Goel (1992) indique que pour ce type de problème, l'état initial, les états désirés, les fonctions d'évaluation ainsi que les fonctions de transformation d'un état à un autre sont bien définis. Pour justifier les différents aspects de leur théorie de résolution de problèmes, Newell et Simon (1972) se sont appuyés sur des problèmes bien structurés, notamment des problèmes de cryptarithme, des problèmes de logique symbolique et le jeu d'échecs. Or, Goel (1992) indique que les problèmes bien structurés sont loin d'être fréquents dans le quotidien. En

référant à la tâche de conception comme une tâche de résolution de problèmes, Goel (1992) tout comme Dorst (2006) indiquent que les problèmes mal définis sont plus fréquents dans la vie de tous les jours.

3.1.3. Environnement de la tâche

L'environnement de la tâche, selon Newell et Simon (1972), est composé des caractéristiques perçues de l'environnement global à un moment particulier en relation avec un but à atteindre, un problème à résoudre ou une tâche à accomplir. Par exemple, lors d'une partie d'échecs, les caractéristiques de chaque pièce permettant de les identifier, la couleur des cases et la position de chaque pièce sont des stimuli visuels pertinents à la tâche. Toutefois, la dimension de l'échiquier et le matériel avec lequel il est construit ne sont pas des informations faisant partie de l'environnement de la tâche même si elles font partie de l'environnement global.

L'environnement de la tâche impose des exigences sur la manière d'atteindre un but. Une exigence est définie par Newell et Simon (1972, p. 79) comme étant une contrainte influençant le comportement du solutionneur de problème qui doit être satisfaite pour atteindre le but. Par exemple, aux échecs, la manière dont les pièces peuvent être déplacées sont des exigences de l'environnement de la tâche. Si une personne déplace un pion de cinq cases en diagonale par exemple, cette personne ne joue plus aux échecs. Les caractéristiques de l'environnement qui engendrent ces exigences constituent la structure de l'environnement. Selon Newell et Simon (1972, p. 825), la structure de l'environnement est l'ensemble d'invariants préservés entre traductions de représentations équivalentes, appelées isomorphes, d'un même problème.

Goel (1992) indique qu'il existe des différences entre l'environnement de la tâche de problèmes bien structurés et l'environnement de la tâche de problèmes mal structurés.

Premièrement, la nature des contraintes est différente. Dans un problème bien structuré comme ceux présentés par Newell et Simon (1972) portant sur des jeux, les contraintes sont logiques, c'est-à-dire que rien n'empêche une personne de violer une contrainte; elle ne joue simplement plus au jeu dans ce cas. Toutefois, pour les problèmes mal structurés en général, les contraintes sont de nature nomologique et/ou intentionnelle. Une contrainte nomologique, aussi appelée causale, est associée aux lois de la nature. Goel (1992) donne comme exemple une poutre qui doit supporter une pression descendante de *x* psi doit exercer une pression égale ou supérieure vers le haut. Ce type de contrainte n'est pas négociable. Quant à une contrainte de nature intentionnelle, elle est régie par l'intervention humaine. Par exemple, la planification financière est un domaine dit intentionnel (Billet et Morineau 2005). Les contraintes sont fixées par des individus et peuvent être négociables. Deuxièmement, la taille et la complexité varient pour les problèmes bien définis et les problèmes mal définis. Ces derniers nécessitent en général plusieurs jours, plutôt que quelques minutes, pour être solutionnés. Troisièmement, la solution d'un problème bien défini peut facilement être évaluée comme bonne ou mauvaise. Quant à un problème mal défini, il est difficile de dire qu'une solution est mauvaise, mais il est possible de dire qu'une solution est meilleure qu'une autre.

Tel que mentionné précédemment, l'individu en tant que STI utilise une mémoire externe. Celle-ci est composée des artéfacts de l'environnement de la tâche. Selon Norman (1993), un artéfact cognitif, tel que présenté à la figure 3.3, consiste en une représentation d'une partie ou de la totalité de l'environnement sur lequel l'individu doit agir. Cette représentation de l'environnement n'est jamais identique à celui-ci car seuls les éléments pertinents à la tâche doivent être représentés.

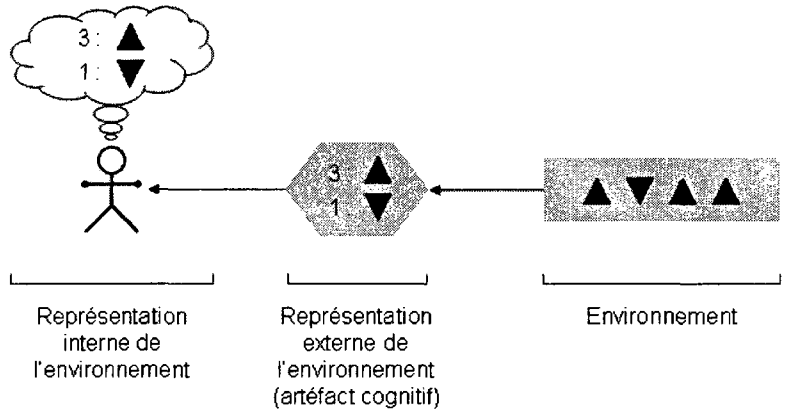


Figure 3.3 – Représentation de l'environnement à travers un artéfact cognitif.

Certains environnements, comme une forêt, sont perceptibles. Dans ce cas, l'environnement de la tâche est constitué en totalité ou en partie de l'environnement sur lequel l'individu doit agir. Il est possible que ce dernier fasse appel à des artéfacts cognitifs, mais ils ne constitueront jamais la totalité de l'environnement de la tâche. Par exemple, un individu qui doit planter des arbres à des endroits spécifiques va utiliser la structure de l'environnement perçu pour déterminer où planter chaque arbre. Il peut également s'aider d'une carte géographique (artéfact cognitif), mais celle-ci sera utilisée en concordance avec l'environnement sur lequel cet individu doit agir.

À contrario, certains environnements ne sont pas perceptibles soit parce qu'il est nécessaire de garder une distance avec ceux-ci, comme un réacteur nucléaire, soit parce qu'ils ne sont pas constitués de composants physiques, comme un portefeuille d'actifs financiers. Dans ce cas, l'environnement de la tâche est constitué en totalité d'artéfacts cognitifs chargés de représenter l'environnement sur lequel l'individu doit agir.

À l'ère de la Net-économie, les systèmes informatiques constituent la principale source d'artéfacts cognitifs. Ces derniers peuvent se présenter sous la forme de systèmes informatiques distribués qui sont composés de plusieurs ordinateurs centraux accessibles

par le Web, d'ordinateurs de bureau ou d'assistants numériques personnels. Quelle que soit sa forme, un système informatique peut être considéré comme étant une extension du cerveau humain en ce sens qu'il réalise plus rapidement et plus efficacement plusieurs de ses fonctions telles le stockage, le traitement et la diffusion d'informations. C'est à travers l'IU que l'individu arrive à interagir avec ce type d'artéfact cognitif. Dans un contexte de résolution de problèmes, le rôle de l'IU consiste donc, d'une part, à présenter les informations décrivant l'état de l'environnement du domaine d'application et, d'autre part, à fournir des mécanismes permettant de modifier celui-ci.

Afin de décrire la nature de la relation entre un utilisateur et une IU, Vicente (1999b) distingue les environnements correspondants des environnements cohérents tels qu'illustrés à la figure 3.4. Un environnement correspondant possède une réalité extérieure à la dyade individu-ordinateur qui impose des contraintes dynamiques de l'environnement sur le comportement d'un individu pour atteindre son but. Les contraintes peuvent donc être de nature nomologique ou intentionnelle. Par exemple, pour un gestionnaire de portefeuille, cette réalité externe comprend les objectifs de placement, la réglementation, les mécanismes des marchés financiers, etc. Le gestionnaire de portefeuille n'est pas en contrôle parfait de l'environnement; il doit s'adapter à celui-ci.

Quant à un environnement cohérent, il n'est assujetti à aucune contrainte environnementale devant être respectée. Vicente (1999b) donne l'exemple d'un logiciel de traitement de texte; cet environnement n'a pas de réalité dynamique, physique ou sociale, extérieure à la dyade individu-ordinateur devant être prise en compte par l'individu pour atteindre son but.

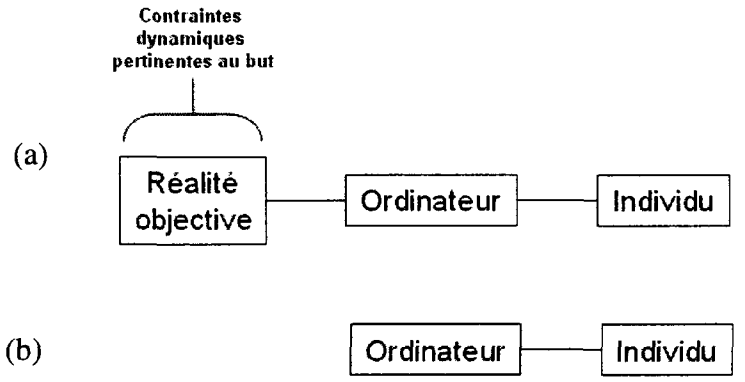


Figure 3.4 – (a) Environnement correspondant et (b) environnement cohérent [adaptée et traduite de Vicente (1999b)].

La distinction entre les environnements correspondants et les environnements cohérents correspond à la distinction entre, respectivement, travailler *par l'entremise* d'un ordinateur et travailler *avec* un ordinateur selon Vicente (1999b). Le point de vue adopté est important dans la mesure où le concepteur d'IU s'affaire à optimiser l'interaction avec l'ordinateur ou avec l'environnement par le biais de l'ordinateur. Les tâches de résolution de problèmes portent généralement sur des environnements correspondants. Ainsi, l'IU doit être en mesure de représenter cette réalité externe objective qui est nulle autre que la structure de l'environnement telle que définie précédemment.

3.1.4. Espace de problème

L'espace de problème est l'endroit où se fait la résolution de problèmes. Il consiste en une représentation interne plus ou moins complète de l'état actuel de l'environnement externe, de son état désiré, des états intermédiaires et des possibilités d'actions pour atteindre le but. Newell et Simon (1972) présentent l'espace de problème d'une manière conceptuelle sans indiquer les composants du cerveau qui sont en cause.

La figure 3.5 illustre la relation entre l'environnement, l'environnement de la tâche et l'espace de problème. Tel que mentionné précédemment, l'environnement de la tâche est

un sous-ensemble de l'environnement global qui ne comprend que les éléments essentiels à la tâche. L'espace de problème est, quant à lui, façonné à partir de l'environnement de la tâche.

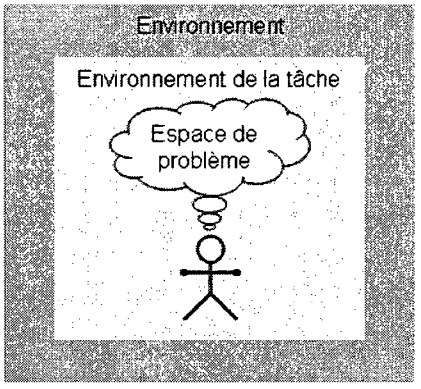


Figure 3.5 – Environnement, environnement de la tâche et espace de problème.

Selon Newell et Simon (1972, p. 823), la structure de l'espace de problème est largement déterminée par la structure de l'environnement de la tâche. Ils indiquent toutefois que la représentation de l'environnement de la tâche dans l'espace de problème n'est pas nécessairement la même d'un individu à l'autre. À ce sujet, Tversky et Kahneman (1981) ont démontré que le cadrage d'un problème, c'est-à-dire la manière dont il est présenté, a un impact sur la décision du sujet. Cependant, selon Newell et Simon (1972), un ensemble d'invariants, qui constitue la structure de l'environnement, est préservé dans chacune de ces représentations. Cette structure impose des contraintes sur l'ensemble des actions possibles pour réaliser la tâche. Il sera possible d'atteindre un but fixé seulement si les actions respectent les contraintes de l'environnement.

Comment est obtenu cognitivement cet ensemble d'actions possibles? Newell et Simon (1972) utilisent le terme *méthode* pour faire référence à des structures d'informations procédurales indiquant comment modifier l'environnement. Ils présentent plusieurs

scénarios décrivant comment sont utilisées ces méthodes pour arriver aux buts, par contre, ils restent vagues sur la manière d'obtenir celles-ci. Une réponse à cette question peut se trouver en se tournant vers la psychologie écologique. Cette école de pensée postule qu'il existe une réciprocité entre les organismes et leur environnement. Cette réciprocité provient du fait que l'environnement influence les organismes dans leur manière d'arriver à leurs buts et que l'atteinte de ces buts implique une modification de l'environnement. Cette réciprocité va dans le même sens que la théorie de résolution de problèmes par l'humain de Newell et Simon (1972). À cet effet, Gibson (1979) propose la théorie de l'affordance. En somme, celle-ci postule que les propriétés perçues de l'environnement ou de ses objets permettent un ensemble d'actions en relation avec un but à atteindre. Par exemple, les propriétés d'une pierre font en sorte qu'on peut l'utiliser pour tenir des livres, enfoncer un pieu ou décorer un jardin, mais pas pour se nourrir ou se vêtir. Ces propriétés proviennent de la structure de l'environnement. Quel que soit le but recherché, les propriétés resteront toujours les mêmes, mais elles pourront être utilisées pour atteindre différents buts.

3.1.5. Synthèse

Un individu fait face à un problème lorsqu'il ne connaît pas la séquence d'actions lui permettant d'arriver à son but. Newell et Simon (1972) ont proposé une théorie de résolution de problèmes décrivant les processus cognitifs d'un individu à ce sujet. Cette théorie postule que, pour comprendre la manière dont un individu résout un problème, il est essentiel de connaître l'environnement pour lequel la tâche de résolution de problèmes est réalisée. Cette description de l'environnement consiste à en décrire la structure, c'est-à-dire les contraintes devant être respectées par la séquence d'actions choisie.

La représentation externe de cet environnement perçu par l'individu est appelée l'environnement de la tâche. De nos jours, l'environnement de la tâche est constitué presqu'exclusivement de technologies de l'information. C'est donc par l'intermédiaire d'une IU que l'individu résout des problèmes. Il est donc essentiel que les IU présentent de manière explicite la structure de l'environnement afin de soutenir l'individu dans la tâche de résolution de problèmes.

3.2. Qu'est-ce qu'une interface écologique?

Une IE permet de soutenir le processus de résolution de problèmes de l'utilisateur en rendant explicite la structure du domaine de travail. Contrairement aux IU traditionnelles qui, afin de représenter l'état d'un environnement spécifique, mettent l'accent sur la présentation d'informations, une IE met plutôt l'accent sur les relations entre ces informations tout en présentant celles-ci.

La conception d'IE s'appuie sur un cadre théorique qui tire ses origines de la psychologie écologique. Cette école de pensée en psychologie postule, entre autres, qu'il n'est pas possible de comprendre le comportement de l'être humain sans comprendre au préalable l'environnement, défini en termes de contraintes, dans lequel celui-ci agit. Selon Gibson (1979), le raisonnement de tout être vivant est façonné par les propriétés de l'environnement dans lequel celui-ci doit agir. Sa perception de l'environnement est constituée de contraintes qui circonscrivent les actions pouvant être réalisées pour atteindre un but. En s'appuyant sur cette prémisse, Vicente et Rasmussen (1992) ont posé les fondations de la conception d'IE.

3.2.1. Types d'événements

Selon Vicente et Rasmussen (1992) un individu peut faire face à trois types d'événements dans son travail :

- **Familiers.** Ces événements font partie de la routine de l'individu; ils sont fréquemment rencontrés. Par conséquent, l'individu peut, en principe, faire face à ces événements sans problème.
- **Occasionnels, mais anticipés.** Ces événements surviennent peu fréquemment. Toutefois, comme ils ont été anticipés par les concepteurs, une panoplie d'aides est disponible à l'individu pour faire face à ces événements.
- **Imprévus.** Comme la catégorie précédente, ces événements surviennent en général peu fréquemment, mais n'ont pas été prévus par les concepteurs. Par conséquent, ils requièrent une certaine improvisation de la part de l'individu. Celui-ci est donc en situation de résolution de problème.

Vicente (2002) mentionne qu'un individu peut faire face aux deux premières catégories d'événements à l'aide d'une séquence d'actions bien définies qui, dans bien des cas, peut être exprimée sous la forme d'un ensemble de règles ou d'algorithmes. Toujours selon lui, ces actions sont de plus en plus réalisées par des systèmes informatiques. Pour certains types d'environnements, appelés systèmes sociotechniques complexes par Vicente (1999a), les événements imprévus ne peuvent être évités et peuvent être accompagnés de conséquences désastreuses lorsqu'ils surviennent (Vicente et Rasmussen 1992). Une IE vise donc à soutenir le raisonnement d'un individu lorsque celui-ci fait face à des événements imprévus qui amènent l'environnement dans un état indésirable. L'objectif de l'individu à ce moment est de ramener celui-ci à un état convenable. En d'autres mots, le rôle de l'individu passe d'un agent d'exécution à un agent d'adaptation.

3.2.2. Les niveaux de contrôle cognitif

Comment soutenir l'individu lors d'événements imprévus? Afin de répondre à cette question, il importe de décrire les niveaux de contrôle cognitif, c'est-à-dire les différentes manières dont l'individu traite l'information perçue de son environnement afin d'atteindre un but désiré. À cet effet, le cadre théorique derrière les IE s'appuie sur la taxonomie SRK de Rasmussen (1983). Celle-ci consiste en un modèle qualitatif représentant trois niveaux inter-reliés de contrôle cognitif tel qu'illustré à la figure 3.6. Chaque niveau correspond à un type particulier de représentation interne de l'environnement : comportement fondé sur les habiletés (*Skill-based behaviour,* SBB), comportement fondé sur les règles (*Rule-based behaviour,* RBB) et comportement fondé sur les connaissances (*Knowledge-based behaviour,* KBB). Le recours à un niveau plutôt qu'à un autre est déterminé par la manière dont un individu interprète une information perçue. La même information peut être interprétée comme étant un signal, un signe ou un symbole.

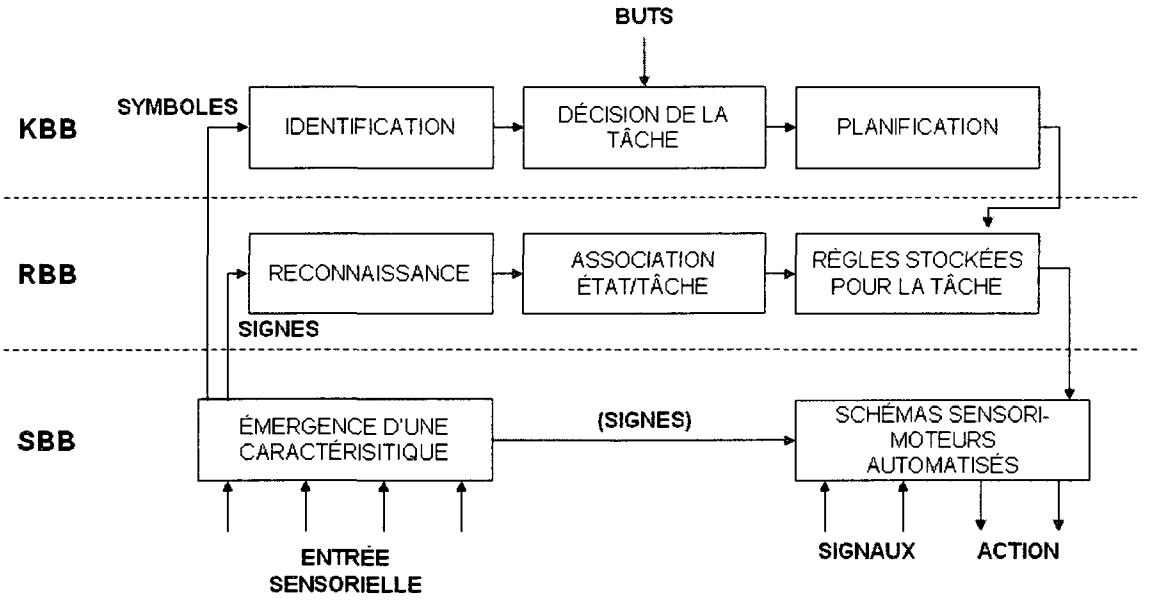


Figure 3.6 – La taxonomie SRK [adaptée et traduite de Rasmussen (1983)].

Les signaux, associés au niveau SBB, consistent en des variables spatio-temporelles physiques représentant le comportement de l'environnement. Ce sont, en d'autres mots, des données sensorielles perçues permettant l'activation de schémas sensori-moteurs automatisés. À ce niveau, le travail réalisé s'appuie sur un contrôle par *proaction*[12] se rapportant à la coordination de mouvements rapides du corps. Quel que soit le niveau de contrôle cognitif activé (SBB, RBB ou KBB), c'est toujours au niveau SBB que les actions sont réalisées sur l'environnement par l'individu.

Lorsque l'information ne permet pas une action directe sur l'environnement, celle-ci peut être interprétée comme étant un signe. Les signes, propres au niveau RBB, sont associés à certaines caractéristiques de l'environnement suggérant un ensemble de tâches prédéfinies. Ils ne représentent en aucun cas des propriétés fonctionnelles de l'environnement; leur seule utilité est la sélection des règles contrôlant la séquence de sous-routines mises en application au niveau SBB. À ce niveau de contrôle cognitif, l'individu n'a pas besoin de s'appuyer sur une compréhension de l'environnement. Il ne fait qu'appliquer les tâches prescrites dans des situations précises.

À la figure 4.1, une flèche avec l'étiquette « signes » entre parenthèses relie les deux rectangles du niveau SBB. Est-ce à dire que les signes peuvent être associés au niveau SBB? Tel que défini au paragraphe précédent, les signes sont propres au niveau RBB. Toutefois, Vicente (1999a, pp. 286-287), en adoptant une perspective temporelle, distingue les activités diachroniques des activités synchroniques comme étant associées au traitement des signes. Les activités diachroniques requièrent une évaluation ou une planification s'appuyant sur des expériences passées pour interpréter l'environnement en

[12] Traduction libre du terme anglais *feedforward*. Le terme *feedback* est généralement traduit par le terme *rétroaction*. Le préfixe *rétro*, qui vient du grecque, signifie *en arrière*. L'antonyme de ce préfixe est *pro* qui signifie *en avant*.

termes de signes pouvant déclencher des actions au niveau SBB dans un futur rapproché. Pour ce type d'activités, les signes sont donc traités au niveau RBB de la manière décrite au paragraphe précédent. Quant aux activités synchroniques, elles surviennent en temps réel et portent également sur des signes activant des actions au niveau SBB, mais en s'appuyant sur la connaissance de la situation présente. Vicente (1999a, p. 286) donne l'exemple d'un conducteur qui change de vitesse après avoir entendu un ton particulier associé à l'augmentation du nombre de tour du moteur. C'est pour cette raison que l'étiquette « signe » a été utilisée au niveau SBB, même si, dans les faits, elle se rapporte au niveau RBB.

Si l'information n'est ni interprétée comme un signal ou un signe, il s'agit donc d'un symbole. Les symboles sont utilisés au niveau KBB dans un processus de raisonnement analytique permettant de prédire ou d'expliquer un comportement de l'environnement. Contrairement aux signaux et aux signes qui portent sur une description physique des caractéristiques de l'environnement, les symboles font appel au sens de l'information perçue. Concrètement, le contrôle cognitif exercé au niveau KBB permet de créer un ensemble de règles et de tâches, en fonction d'un but à atteindre, permettant de faire face à la situation actuelle pour laquelle il n'existe présentement aucune règle d'exécution stockée. Le résultat de ce processus sera envoyé au niveau RBB pour y être stocké et ensuite réalisé au niveau SBB.

Généralement, un individu a recours aux niveaux SBB et RBB lorsqu'il fait face à des événements familiers ou occasionnels et a recours au niveau KBB lorsqu'il se retrouve devant des événements imprévus. Toutefois, Vicente (1999a, p. 288) mentionne qu'un individu peut recourir au niveau KBB même devant un événement prévu. De plus, il mentionne qu'une tâche ne peut être associée à un niveau de contrôle cognitif spécifique. Selon lui, trois facteurs viennent influer sur le niveau de contrôle cognitif

utilisé par l'individu : (a) son niveau d'expertise, (b) la manière dont les informations sont présentées à travers l'IU et (c) le degré de réflexion de l'individu sur son travail.

3.2.3. Principes de conception

Selon Vicente et Rasmussen (1992), une IE ne doit pas imposer un niveau de contrôle cognitif supérieur à celui requis par la tâche tout en soutenant les trois niveaux de contrôle cognitif à la fois. Afin d'atteindre cet objectif, chaque niveau de contrôle cognitif est associé à un principe de conception spécifique (Vicente et Rasmussen 1992, Vicente 1999a, pp. 295-296, 317-326).

Au niveau SBB, les informations perçues sont interprétées comme des signaux nécessaires à la manipulation des différents objets du domaine de travail. En s'appuyant sur le paradigme d'interface à manipulation directe (Schneiderman 1982), **l'individu doit pouvoir manipuler les objets du domaine de travail directement sur l'IU**. Celle-ci utilise donc un mode de présentation plutôt graphique qu'alphanumérique. La présentation graphique s'appuie sur une représentation des objets du domaine de travail à l'aide de formes géométriques inter-reliées. Ceci n'exclut pas complètement le texte de l'IU, mais sa présence est limitée au minimum nécessaire. Cette forme de présentation favorise une visualisation de la dynamique du système à travers une représentation des informations sous forme de signaux spatio-temporels, impliquant ainsi un traitement cognitif perceptuel plutôt qu'analytique.

Au niveau RBB, les informations perçues sont interprétées comme des signes permettant de déclencher la procédure appropriée. **Une correspondance un-à-un doit exister entre les contraintes du domaine de travail et les informations présentées par l'IU**. La structure du domaine de travail peut être considérée comme un système de termes inter-reliés. Cette structure sous-entend un ensemble de contraintes qui se traduit par des

relations spécifiques entre plusieurs termes. Les indices suggérant l'action peuvent généralement se présenter sous forme d'écarts de valeur par rapport à une cible établie. Dépendamment des éléments en cause, des règles spécifiques portant sur ces éléments guident l'individu dans les tâches à entreprendre pour résoudre la situation.

Au niveau KBB, les informations perçues sont interprétées comme des symboles nécessaires à la résolution de problèmes. Pour soutenir l'individu à ce niveau de contrôle cognitif, **l'IU doit présenter la hiérarchie fonctionnelle du domaine de travail**, c'est-à-dire sa structure à travers différents niveaux d'abstraction. Selon Vicente et Rasmussen (1992), cette représentation externe est psychologiquement compatible avec la représentation interne du domaine de travail réalisée par un individu puisqu'elle fait le lien entre les buts recherchés (niveau supérieur de la hiérarchie) et l'environnement qui doit être manipulé (niveau inférieur de la hiérarchie).

Ces trois principes de conception s'imbriquent les uns dans les autres. En effet, l'interface à manipulation directe doit représenter tous les objets du domaine de travail et les relations entre celles-ci. Des ensembles spécifiques de relations entre les termes du domaine de travail constituent les contraintes de celui-ci. Ces contraintes s'appuient sur une structure du domaine de travail qui, lorsque hiérarchisée par niveau d'abstraction, constitue la hiérarchie fonctionnelle du domaine de travail.

La figure 3.7 compare une IU traditionnelle[13] et une IE qui applique ces principes pour le contrôle d'un processus thermo-hydraulique. L'objectif de ce système est, d'une part, de répondre à la demande externe de quantité d'eau de chaque réservoir et, d'autre part,

[13] L'IU traditionnelle s'appuie sur le paradigme de « capteur unique, indicateur unique ». En d'autres mots, les concepteurs présentent principalement les informations captées physiquement et non les informations d'ordre fonctionnel, c'est-à-dire pouvant être dérivées des premières. Il s'agit d'une tendance forte dans les domaines du contrôle de processus industriel (Goodstein 1981) et de la santé (Sharp et Helmicki 1998, Miller 2000).

de garder la température de l'eau à 40°C pour le réservoir 1 et à 20°C pour le réservoir 2. Il est à noter que la demande externe d'eau peut varier à tout moment (événements imprévus). Pour atteindre cet objectif, deux circuits redondants d'écoulement d'eau permettent d'acheminer l'eau à un ou aux deux réservoirs. L'opérateur contrôle huit valves (VA, VA1, VA2, VO1, VB, VB1, VB2 et VO2), deux pompes (PA et PB) et deux brûleurs (HTR1 et HTR2).

Ce qui distingue les deux IU de la figure 3.7 est l'ensemble des informations présentées portant sur ce domaine de travail. Dans le cas de l'IU traditionnelle, les informations portent uniquement sur l'état des composants physiques. Il s'agit de la température de l'eau à l'entrée du système (T0), de la température de l'eau à la sortie des réservoirs (T1 et T2) et du volume de chaque réservoir (V1 et V2). Quant à l'IE, celle-ci présente, en plus des informations de l'IU traditionnelle, des informations fonctionnelles, c'est-à-dire qui portent sur l'état des fonctions que tente d'atteindre l'opérateur à l'aide des composants physiques. Ces informations fonctionnelles portent sur les flux d'eau (FVA, FVB, FA1, FA2, FB1, FB2), sur la masse et l'énergie entrantes et sortantes des réservoirs (MI1, MO1, EI1, EO1, MI2, MO2, EI1 et EO2) et sur la relation entre le volume, l'énergie et la température.

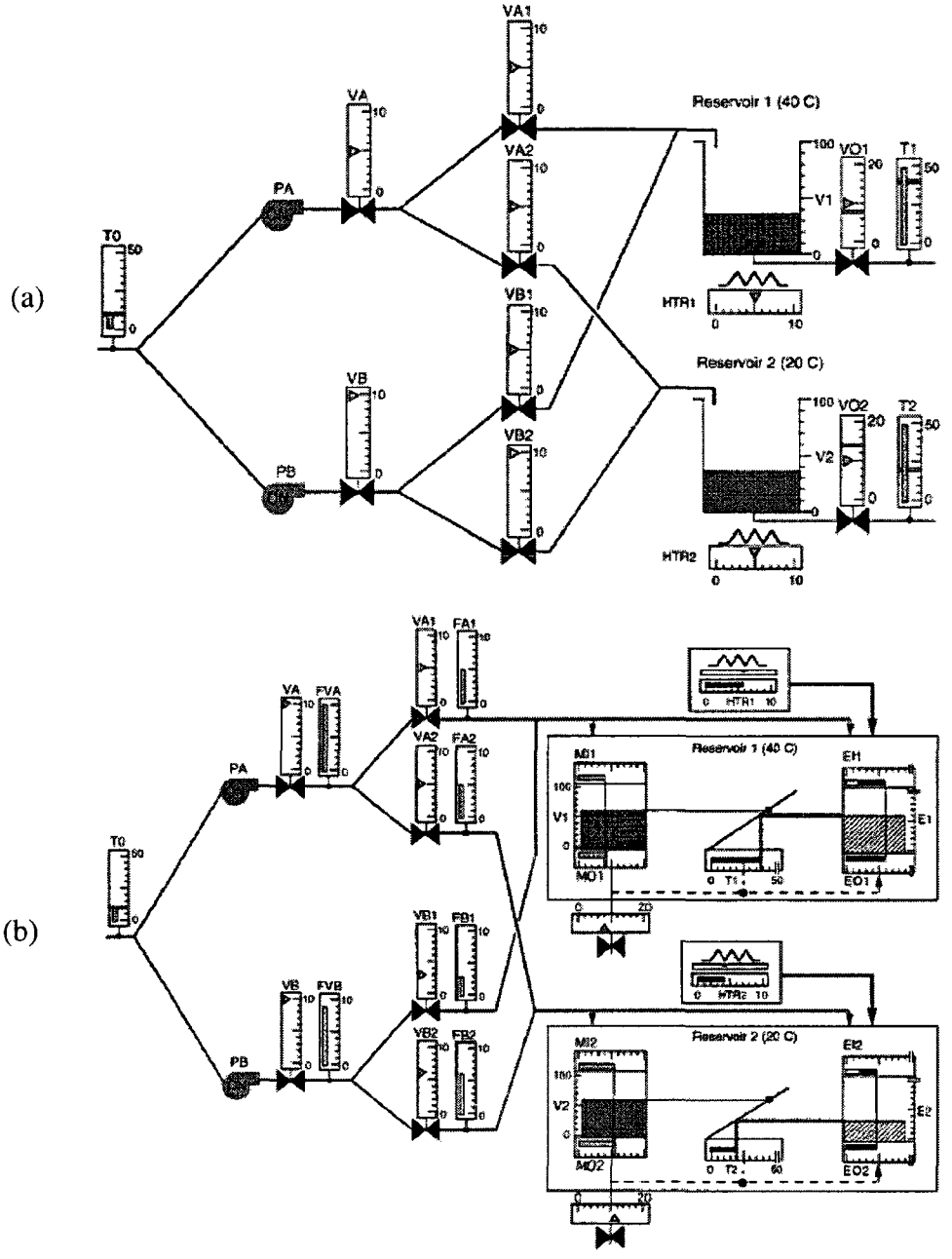


Figure 3.7 – Interface utilisateur traditionnelle (a) et interface écologique (b) pour le contrôle d'un processus thermo-hydraulique (Pawlak et Vicente 1996).

Jusqu'à quel degré les deux IU présentées à la figure 3.7 soutiennent-elles les trois niveaux de contrôle cognitif? Premièrement, les deux IU sont à manipulation directe; l'utilisateur, à l'aide d'une souris, peut directement manipuler les objets du domaine de

travail, par exemple, activer ou désactiver une valve ou une pompe ou régler la température d'un brûleur. Elles soutiennent donc le niveau SBB. En ce qui concerne le niveau RBB, les contraintes se manifestent par des indicateurs d'écarts. Seule l'IE comporte de tels indicateurs. Ces derniers se trouvent à l'intérieur des rectangles entourant chacun des deux réservoirs. Des lignes indiquent la valeur cible à atteindre pour la température et le volume d'eau. Quant au niveau KBB, seule l'IE le supporte. En effet, l'IU traditionnelle présente des informations portant uniquement sur les aspects physiques du domaine de travail et l'IE présente des informations portant non seulement sur les aspects physiques, mais également fonctionnels. Par exemple, si les valves VA et VA1 sont ouvertes au maximum et que le flux FVA est à pleine capacité, mais le flux FA1 est à mi-capacité, l'opérateur peut clairement voir qu'il y a un problème avec la valve VA1. L'opérateur ne peut tirer directement cette conclusion avec l'IU traditionnelle. En somme, toutes les informations de la hiérarchie fonctionnelle sont présentées dans l'IE.

Puisque l'IE doit présenter toutes les informations pertinentes à la représentation de la structure et des contraintes du domaine de travail, celle-ci contient un nombre plus élevé d'informations que l'IU traditionnelle. Néanmoins, lors d'une étude expérimentale, cette IE a contribué à une détection plus rapide d'anomalies et un diagnostic plus précis par rapport à l'IU traditionnelle (Pawlak et Vicente 1996). Vicente, Christoffersen et Pereklita (1995) apportent une explication au succès de cette IE. Les résultats d'une de leurs études expérimentales démontrent que les sujets ayant obtenu les meilleures performances pour le diagnostic avaient tendance à commencer leur recherche du problème à un niveau d'abstraction élevé et procéder graduellement vers les niveaux inférieurs qui contiennent plus de détails. En somme, pour soutenir la résolution de problèmes, c'est-à-dire le niveau KBB de contrôle cognitif, il est essentiel d'afficher autant les informations qui portent sur les aspects physiques que fonctionnels du

domaine de travail. Toutefois, Burns (2000) apporte une précision à cette conclusion en commentant les résultats de son étude expérimentale. Elle indique qu'il ne s'agit pas simplement de présenter plus d'informations (physiques et fonctionnelles), mais de le faire de manière à ce que les relations entre celles-ci soient explicites.

3.2.4. Éléments visuels d'une interface écologique

D'après les trois principes de conception présentés à la section précédente, afin de soutenir les trois niveaux de contrôle cognitif, une IE doit présenter les *objets* du domaine de travail qui est sous le contrôle de l'utilisateur, les *contraintes* de ce domaine de travail et la *hiérarchie fonctionnelle* sous-jacente. Toutefois, ces trois principes restent vagues dans la mesure où ils n'indiquent pas précisément les éléments que le concepteur doit manipuler pour créer une IE. À ce sujet, Burns et Hajdukiewicz (2004) présentent un thésaurus visuel qui offre un catalogue de composants visuels réutilisables, c'est-à-dire des agencements particuliers de formes géométriques s'appliquant à des types spécifiques de relations entre des termes. En d'autres mots, il s'agit de patrons visuels de présentation de relations entre des termes. Cette section vise à présenter ce thésaurus visuel et comprendre comment il s'applique à la représentation des objets, des contraintes et de la hiérarchie fonctionnelle.

3.2.4.1. Patrons de conception d'interfaces utilisateurs

La littérature attribue à Alexander, Ishikawa, Silverstein, Jacobson, Fiksdahl-King et Angel (1977) la définition d'un patron de conception. Il s'agit d'un agencement d'éléments permettant de solutionner un problème de conception qui tend à se répéter dans un contexte particulier. Initialement, les patrons de conception ont été appliqués en architecture. Toutefois, durant les années 1990, le domaine du génie logiciel s'est approprié ce concept afin de soutenir la conception de systèmes informatiques orientés objets. Notamment, l'ouvrage de Gamma, Helm, Johnson et Vlissides (1995),

incontestablement le plus reconnu sur le sujet, présente différents agencements de classes s'appliquant dans des contextes particuliers.

En ce qui concerne la conception d'IU, certains ouvrages tels Borchers (2001), Graham (2002) et Tidwell (2005) présentent un ensemble de patrons de conception destinés à soutenir l'interaction dans différents contextes. Ces derniers associent un agencement d'éléments visuels à des situations spécifiques pouvant être rencontrées par l'utilisateur, par exemple indiquer le progrès d'un traitement, offrir une vue globale et détaillée à la fois ainsi que présenter un panier d'achat virtuel. Toutefois, aucun de ces patrons de conception n'est destiné à soutenir directement la résolution de problèmes comme c'est le cas avec le thésaurus visuel de Burns et Hajdukiewicz (2004).

3.2.4.2. Patrons de conception d'interfaces écologiques

Quels sont donc les éléments visuels d'une IE? Le thésaurus visuel de Burns et Hajdukiewicz (2004) fait état de termes et de relations représentant presqu'exclusivement des équations mathématiques. Les relations entre les termes représentent donc des opérateurs mathématiques. En somme, une IE consiste à représenter, d'une manière graphique selon le premier principe de conception, des *expressions* mathématiques associant des *termes* et des *opérateurs* tels qu'illustrés à la figure 3.8. Un terme peut être une variable, c'est-à-dire pouvant prendre plusieurs valeurs numériques, ou une constante, c'est-à-dire pouvant prendre qu'une seule valeur numérique. À la figure 3.8, les termes a et b sont des constantes et les termes x, y et z sont des variables.

Figure 3.8 – Représentations graphiques de relations mathématiques (Burns et Hajdukiewicz 2004).

Burns et Hajdukiewicz (2004) présentent un cas où ces agencements sont utilisés pour représenter une voiture. La pression, le débit d'injection et la température sont des variables associées au moteur. Des contraintes telles le temps limite pour arriver à la destination, la distance maximale pouvant être parcourue considérant la quantité restante de carburant ou le nombre de places disponibles sont représentées graphiquement.

3.2.4.3. Représenter les objets du domaine de travail

Le premier principe de conception d'IE fait référence à la manipulation directe des objets du domaine de travail. La description d'un objet est réalisée par un *concept* qui représente un regroupement logique de termes ou d'autres concepts. Par exemple, dans le cas d'un portefeuille d'actifs financiers, le portefeuille est un concept qui regroupe des variables telles le rendement et le risque. De plus, le portefeuille contient des actifs financiers qui constituent un autre concept regroupant également des variables telles le rendement, le risque, la valeur et la quantité. Le portefeuille est associé à un client, un autre concept qui regroupe des variables telles son nom, son objectif de rendement et son seuil de tolérance au risque. Il est possible d'assigner une valeur à une variable, mais pas à un concept.

L'état d'un objet particulier est représenté par la valeur de chaque terme associée à cet objet à tout moment. Par exemple, un portefeuille a une valeur associée au rendement et

au risque à un moment précis. Ces valeurs constituent l'état du portefeuille à ce moment précis.

La manipulation des objets du domaine de travail passe donc par la manipulation de la valeur des variables qui leur sont associées. Par exemple, pour modifier le rendement d'un actif financier, il faut en modifier sa quantité. Pour ce faire, il faut procéder à l'achat et la vente de cet actif financier. L'objet (actif financier) en tant que tel n'est pas manipulé, mais plutôt une partie de celui-ci l'est (quantité). Par conséquent, les variables associées aux concepts peuvent être de deux types : celles dont la valeur peut être modifiée suite à une manipulation par l'opérateur ou un changement dans l'environnement et celles qui subissent l'influence des variables du premier type.

3.2.4.4. Représenter des contraintes du domaine de travail

Le deuxième principe de conception d'IE fait référence aux contraintes du domaine de travail. Une contrainte permet de savoir si le système sous le contrôle de l'opérateur est dans un état normal ou pas. Elles sont généralement représentées sous forme d'écart avec une valeur cible définie par une constante ou une variable. Si la valeur actuelle correspond à la valeur cible, la contrainte est respectée. Dans le cas contraire, il y a bris de contrainte se traduisant en une anomalie du système. Par conséquent, une contrainte peut donc être considérée comme étant une association entre différents termes sous la forme d'une expression logique, c'est-à-dire reliées à l'aide d'un opérateur logique. À la figure 3.8, la représentation géométrique de gauche correspond à une contrainte. Dans le cas du portefeuille d'actifs financiers, le rendement du portefeuille qui doit être supérieur ou égal à l'objectif de placement du client et le risque du portefeuille qui doit être inférieur au seuil de tolérance au risque du client sont deux contraintes.

3.2.4.5. Représenter la hiérarchie fonctionnelle du domaine de travail

Le troisième principe de conception fait référence à la hiérarchie fonctionnelle. Celle-ci consiste à représenter le domaine de travail à travers différents niveaux d'abstraction. Le niveau le plus élevé est associé aux buts à atteindre et le niveau le moins élevé est associé à l'environnement. Des termes particuliers sont associés à chacun de ces niveaux d'abstraction. De plus, il existe un réseau de termes intermédiaires entre les termes de ces deux niveaux. Chaque concept regroupe donc des termes à différents niveaux d'abstraction. Ce réseau de termes entre différents niveaux constitue la hiérarchie fonctionnelle. Il s'agit de la structure de « buts-moyens » de l'environnement. Pour deux niveaux consécutifs, les termes du niveau supérieur représentent le « pourquoi? » du niveau inférieur, c'est-à-dire le but, et les termes du niveau inférieur représentent le « comment? » du niveau supérieur, c'est-à-dire les moyens.

En ce qui concerne le cas du portefeuille d'actifs financiers, la valeur du rendement d'un actif financier peut être obtenue par la division de la variation de sa valeur (valeur actuelle soustraite de la valeur initiale) par sa valeur initiale. Les variables de cette règle peuvent être associées à trois des cinq niveaux d'abstraction définis par Moïse et Noiseux (2007) pour la gestion de portefeuille d'actifs financiers, du plus élevé au moins élevé : effets (rendement), variations (différence entre valeur actuelle et valeur initiale), indicateurs (valeur initiale et valeur actuelle). Les variables associées à un de ces niveaux représentent les buts du niveau inférieur et les moyens du niveau supérieur. Même si elles sont réparties dans différents niveaux d'abstraction, les variables sont toutes rattachées au même concept, celui d'actif financier.

En somme, la hiérarchie fonctionnelle peut se traduire par un ensemble d'équations mathématiques dont les relations sont représentées par différents opérateurs

mathématiques tels des opérateurs d'agrégation (p. ex. : somme, nombre) ou des opérateurs arithmétiques (p. ex. : addition, multiplication).

3.2.5. Applications et résultats d'expériences

Les IE ont été appliquées à différents domaines notamment le contrôle de processus industriel, l'aviation, la gestion de réseaux informatiques et la médecine (Vicente 2002, Burns et Hajdukiewicz 2004). Jusqu'à présent, les recherches ont démontré la supériorité des IE par rapport à des IU traditionnelles (Vicente 2002), et ce, autant pour des domaines causaux, c'est-à-dire régis par les lois de la nature, qu'intentionnels, c'est-à-dire régis par l'activité humaine (Billet et Morineau 2005). Cette performance se manifeste par une détection plus rapide d'anomalies, un diagnostic plus précis, des stratégies d'action plus sophistiquées ainsi qu'une diminution du temps nécessaire pour accomplir une tâche.

En passant en revue plusieurs études expérimentales, Vicente (2002) a dénoté plusieurs avantages à l'utilisation d'une IE par rapport à une IU traditionnelle. En voici quelques uns :

Une IE permet une meilleure performance lors de scénarios anormaux. En ce qui concerne des scénarios normaux, une IE offre sensiblement la même performance qu'une IU traditionnelle.

Il existe une corrélation entre la performance des sujets et la complexité du domaine de travail. Plus la structure du domaine de travail est complexe, plus l'effet de l'IE est marqué.

Il existe une corrélation inverse entre la performance des sujets et leur expérience par rapport au domaine de travail. Plus les sujets sont experts dans le domaine de travail, moins l'effet de l'IE est marqué.

En somme, la valeur d'une IE réside dans les situations où un utilisateur non-expert doit s'adapter au contexte changeant d'un domaine de travail complexe. Pour les autres situations, la lourdeur d'une IE, en termes de quantité d'informations présentées, ne constitue pas un obstacle à la performance de l'utilisateur. Vicente (2002) attribue ces avantages aux facteurs suivants :

La différence de performance n'est pas attribuée à la forme visuelle de l'IE, mais à son contenu informationnel qui diffère de celui d'une IU traditionnelle.
La présentation d'informations d'une IE s'appuie sur une hiérarchie fonctionnelle de l'environnement.
Une IE capitalise sur les ressources spatiales plutôt que verbales.
Une IE permet un contrôle du domaine de travail à un niveau d'abstraction plus élevé.

3.2.6. Discussion

L'approche de conception d'IE est conforme à la théorie de résolution de problèmes de Newell et Simon (1972), présentée à la section 3.1, sur deux points. Premièrement, cette théorie considère l'individu comme un STI qui traite des symboles stockés en mémoire. En ce qui concerne une IE, sa conception s'appuie sur la taxonomie SRK dont le niveau KBB de contrôle cognitif, associé à la résolution de problèmes, s'appuie sur le traitement de symboles. Deuxièmement, Newell et Simon (1972) indiquent qu'une représentation interne de la structure de l'environnement est essentielle pour soutenir la résolution de problèmes. Une IE, pour soutenir le niveau KBB de contrôle cognitif, doit présenter les informations en s'appuyant sur une hiérarchie fonctionnelle, c'est-à-dire la structure du domaine de travail à travers différents niveaux d'abstraction.

Comment est obtenue la hiérarchie fonctionnelle? La CIE porte principalement sur la manière de concevoir une IE à partir d'une représentation du domaine de travail qui s'appuie sur une hiérarchie fonctionnelle. Les techniques de définition de la hiérarchie fonctionnelle pour un domaine particulier relève donc de l'analyse et non de la conception.

Pourquoi alors une section sur la conception d'IE? Parce qu'une IE représente le produit final, c'est-à-dire un type particulier d'IU permettant de soutenir la résolution de problèmes. C'est à partir d'une description du produit final qu'il est possible d'identifier les éléments que doit manipuler le concepteur d'IE. Ainsi, la technique de représentation du domaine de travail utilisée doit servir à identifier des concepts, des termes et des opérateurs qui serviront à définir la structure du domaine de travail. Pour ce faire, les termes doivent porter sur différents niveaux d'abstraction et être reliées ensemble de manière à obtenir une hiérarchie fonctionnelle du domaine de travail. Le concepteur devra par la suite transformer ces éléments en composants visuels et les disposer sur une surface d'affichage de manière à obtenir une IE.

3.3. Méta-modèle d'une interface écologique

Cette section présente le méta-modèle[14] d'une IE. Il s'agit d'une manière rigoureuse de mettre en relation tous les éléments manipulés par le concepteur d'IE qui sont identifiés

[14] Tous les méta-modèles de cette thèse sont réalisés avec le diagramme de classes du langage UML (OMG 2008b) accompagné du langage OCL (OMG 2006) dans certains cas. Les instances de ces méta-modèles sont réalisées avec le diagramme objet.

Le choix du langage UML repose sur deux raisons. La première est que, en s'appuyant sur la quantité de publications sur le sujet, cette notation semble être bien connue et privilégiée pour la modélisation et la méta-modélisation en génie logiciel. La deuxième raison est que le développement de logiciels, incluant les logiciels de modélisation, est principalement réalisé en abordant le paradigme orienté objet. Or, comme cette notation s'appuie également sur ce paradigme, il existe une correspondance directe entre les éléments représentés et les composants logiciels.

à la section précédente. Les éléments principaux sont les suivants : concept, terme, expression, contrainte et hiérarchie fonctionnelle. Le méta-modèle est divisé en paquetages selon ces derniers.

3.3.1. Concept et terme

La figure 3.9 illustre la section du méta-modèle qui décrit la relation entre les concepts et les termes. Les méta-classes `Concept` et *`Terme`* sont des éléments nommés, c'est-à-dire que chaque instance porte un nom. La méta-classe *`Terme`* est abstraite. Ces instances sont du type `Constante` ou `Variable`. Ces méta-classes héritent des propriétés de la méta-classe *`Terme`*. En plus d'hériter de l'attribut `nom`, elles héritent de l'association `propriété` de la méta-classe `Concept`. L'association entre les méta-classes `Concept` et *`Terme`* représentent une relation de propriété. Une instance de la méta-classe `Concept` est propriétaire d'autres instances de cette méta-classe ou d'instances de la méta-classe *`Terme`*. Inversement, une instance de la méta-classe `Concept` peut être la propriété d'une autre instance de cette même méta-classe et une instance de la méta-classe *`Terme`* doit obligatoirement être la propriété d'une instance de la méta-classe `Concept`.

La syntaxe abstraite aurait pu être représentée à l'aide de n'importe quel langage de modélisation comme par exemple le diagramme entité-association (Chen 1976). L'essentiel est que les éléments d'un méta-modèle et leurs relations entre eux puissent être représentés d'une manière intelligible.

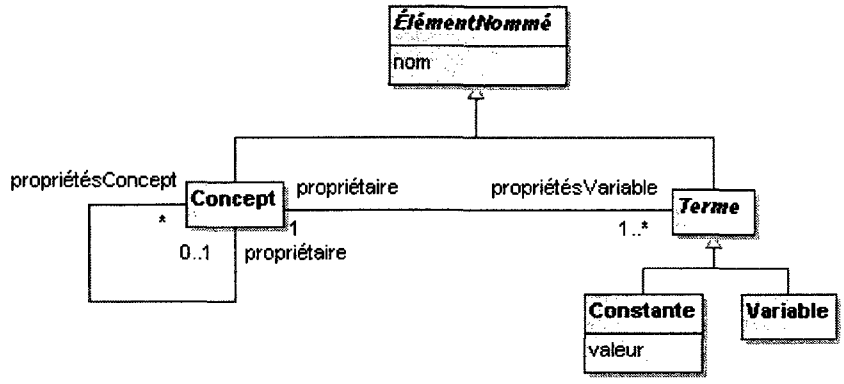


Figure 3.9 – Paquetage des concepts et des termes.

Pour reprendre le cas de la figure 3.7, chaque réservoir est un concept auquel sont rattachées des variables telles la température de l'eau, le volume d'eau, la masse entrante et la masse sortante ainsi que l'énergie entrante et l'énergie sortante. La figure 3.10 illustre quelques instances de ces méta-classes et leur relation avec leur concept propriétaire.

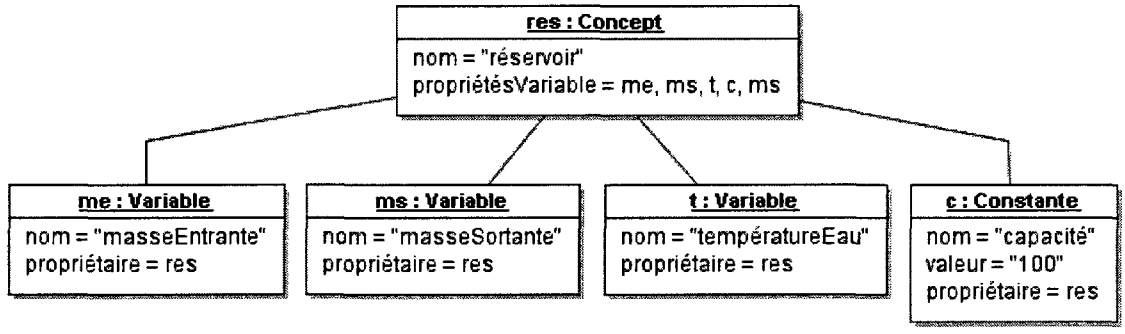


Figure 3.10 – Instances des méta-classes `Concept` et *`Terme`*.

3.3.2. Expression

La figure 3.11 présente le paquetage des expressions[15]. Une expression consiste en un regroupement de termes et d'opérateurs mathématiques. Les opérateurs sont unaires, c'est-à-dire ayant qu'un seul opérande, ou binaires, c'est-à-dire ayant deux opérandes. Un opérateur peut être lui-même un opérande en ce sens que son résultant est considéré comme une variable, donc un terme. C'est pour cette raison que les méta-classes *Terme* et *Opérateur* héritent de la méta-classe abstraite *Opérande*. Pour certains opérateurs binaires tels la division et la soustraction, le sens des opérandes est important. C'est pour cette raison que deux associations obligatoires, nommées respectivement opérandeGauche et opérandeDroite, vont de la méta-classe *Binaire* vers la méta-classe *Opérande*.

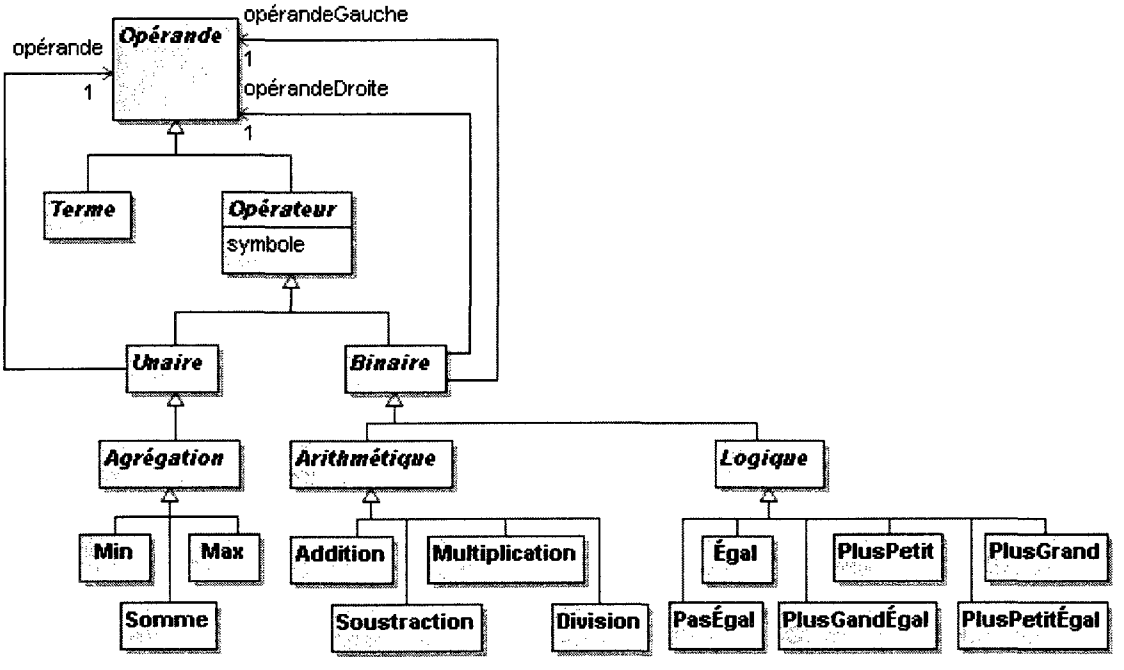
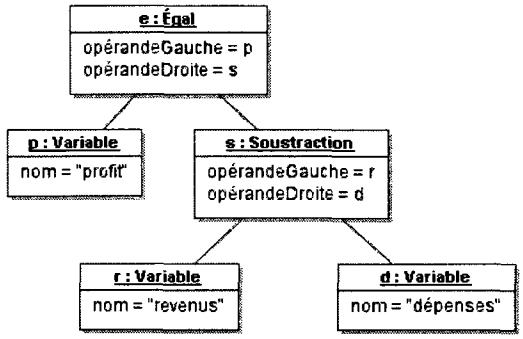


Figure 3.11 – Paquetage des expressions.

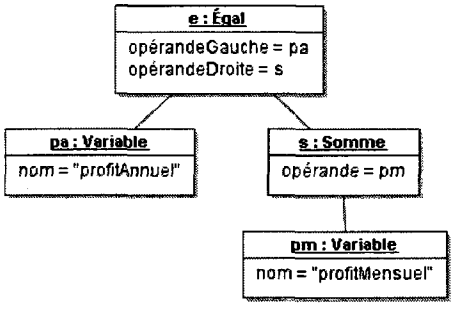
[15] Il est à noter que ce paquetage ne se veut pas exhaustif pour des raisons de simplicité. D'autres types d'opérateurs comme la moyenne ou la variance pourraient être ajoutés.

Il est à noter la figure 3.11 se veut une représentation incomplète du paquetage des expressions pour des raisons de simplicité. Même si des opérateurs comme la moyenne ou la variance n'y apparaissent pas, ils font partie du méta-modèle. L'accent de la figure 3.11 est plutôt mis sur la présentation des trois types d'opérateurs (agrégation, arithmétique, logique) sans présenter tous les opérateurs de ces types.

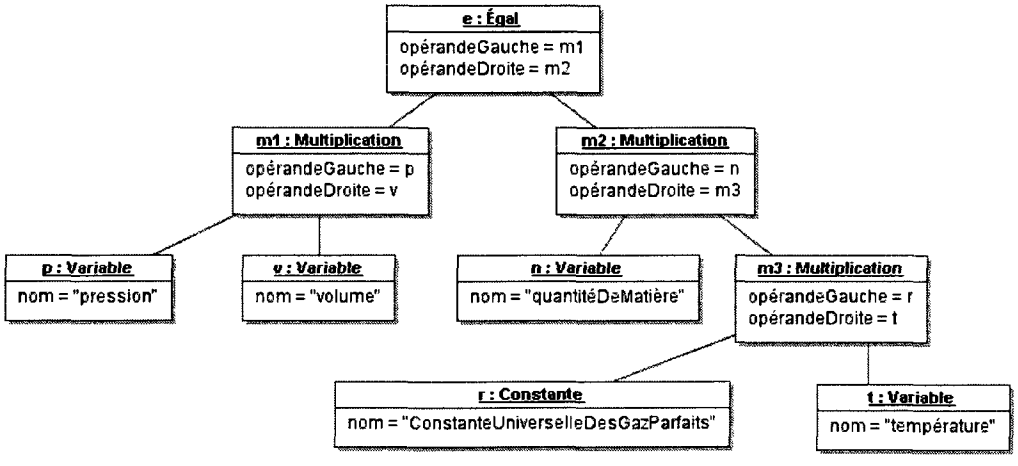
La figure 3.12 présente des expressions mathématiques et leurs instances correspondantes du méta-modèle. Les instances de la méta-classe *Binaire* (Égal, Soustraction, Multiplication) contiennent deux associations, chacune vers un opérande, tandis que l'instance de la méta-classe *Unaire* (Somme) ne contient qu'une seule association vers un opérande.




```
profit = revenus - dépenses
```




```
profitAnnuel = SOMME(profitMenuel)
```




```
PV = nRT
```

Figure 3.12 – Expressions mathématiques et leurs instances correspondantes du méta-modèle.

3.3.3. Contrainte

Tel que mentionné précédemment, une contrainte est représentée par une expression mathématique contenant un opérateur logique. Les trois expressions de la figure 3.12 sont des contraintes; la valeur de la partie de gauche doit être égale à la valeur de la partie de droite, sans quoi la contrainte est brisée. La figure 3.13 présente le méta-modèle des contraintes.

Figure 3.13 – Paquetage des contraintes.

3.3.4. Hiérarchie fonctionnelle

La hiérarchie fonctionnelle est créée par l'utilisation de mêmes termes dans différentes expressions mathématiques. La figure 3.14 présente un ensemble de contraintes et la hiérarchie fonctionnelle sous-jacente. On remarque que la plupart des variables sont utilisées dans plus d'une contrainte. Également, la hiérarchie fonctionnelle impose des niveaux pour chaque contrainte. Par exemple, la contrainte qui porte sur le calcul des revenus et celle qui porte sur le calcul des dépenses sont au même niveau; une n'influe pas l'autre. Ces niveaux représentent la structure de « buts-moyens ». Par exemple, si on désire savoir *comment* faire pour atteindre l'objectif de profitabilité, on voit qu'il faut faire varier les revenus ou les dépenses. Si on désire savoir *pourquoi* on doit faire varier les revenus et les dépenses, on voit que c'est pour atteindre l'objectif de profitabilité.

```
objectif < profits
profits = revenus - dépenses
revenus = prixUnitaire * qtéVendue
dépenses = fraisFixes + fraisVariables
fraisVariables = fraisUnitaire * nbUnitésProduites
```

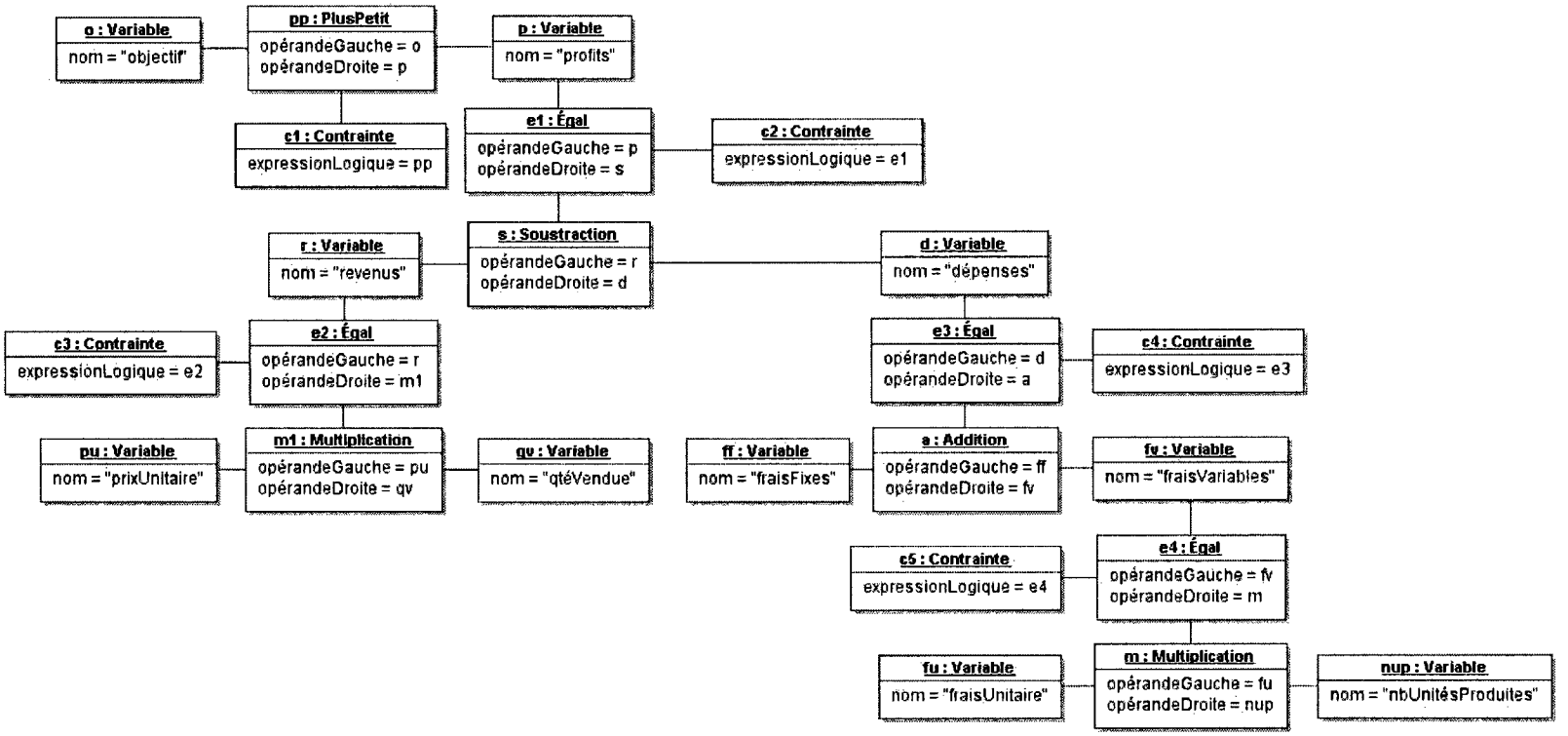


Figure 3.14 – Hiérarchie fonctionnelle sous-jacente à un ensemble de contraintes.

3.4. Conclusion

Ce chapitre présente les IE comme étant un type particulier d'IU soutenant la résolution de problèmes. Plus particulièrement, une IE permet à l'utilisateur de s'adapter lors d'événements imprévus en rendant explicite la hiérarchie fonctionnelle d'un domaine de travail. Cette dernière consiste en une structure de contraintes inter-reliées sur laquelle s'appuie le processus de résolution de problèmes d'un individu.

La CIE consiste à rendre explicite la hiérarchie fonctionnelle à partir d'agencements de formes géométriques. La hiérarchie fonctionnelle est obtenue suite à une analyse du domaine de travail. Le prochain chapitre présente la HAD. Il s'agit de la seule technique de représentation de domaines de travail destiné à la CIE présentée dans la littérature. Le méta-modèle des IE présenté dans ce chapitre est utilisé dans le chapitre suivant pour évaluer la compatibilité entre la HAD et les IE.

CHAPITRE 4 - LA HIÉRARCHIE D'ABSTRACTION ET DE DÉCOMPOSITION

La CIE s'appuie principalement sur une représentation du domaine de travail. La seule technique de représentation à cette fin présentée dans la littérature sur les IE est la HAD. Ce chapitre porte sur la description de cette dernière.

La section 4.1 présente l'analyse du travail cognitif. Il s'agit d'une démarche d'analyse dans laquelle la HAD est la pièce maîtresse. La section 4.2 présente une description de la HAD à partir d'une recension de la littérature. Cette description est ensuite utilisée pour créer un méta-modèle de la HAD qui fait l'objet de la section 4.3. Enfin, ce méta-modèle est utilisé à la section 4.4 pour évaluer la compatibilité entre la structure de la HAD et la structure d'une IE. Cette évaluation permet d'atteindre le premier objectif de cette thèse.

4.1. Analyse du travail cognitif

L'analyse du travail cognitif[16] (Rasmussen, Pejtersen et Schmidt 1990, Rasmussen, Pjetersen et Goodstein 1994, Vicente 1999a) est un cadre d'analyse du travail qui s'appuie sur une approche écologique. Contrairement aux approches centrées sur l'utilisateur qui mettent l'accent sur une analyse de la tâche de ce dernier, l'approche écologique, telle qu'illustrée à la figure 4.1, commence par une analyse du domaine de travail sur lequel celui-ci doit agir pour graduellement arriver à une analyse de l'utilisateur.

[16] Traduction libre du terme anglais *Cognitive Work Analysis*.

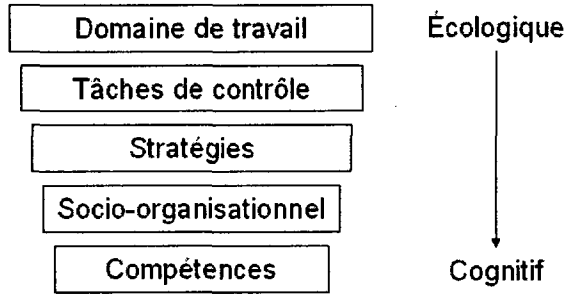


Figure 4.1 – Cadre de l'analyse du travail cognitif [traduite de Vicente (1999a, p. 115)].

L'approche écologique consiste en l'analyse des contraintes imposées par le domaine de travail limitant les actions possibles de l'individu. Les contraintes peuvent être décomposées en cinq niveaux imbriquées où les niveaux supérieurs imposent des limites aux niveaux inférieurs. Le premier niveau, le domaine de travail, porte sur le domaine de travail sous le contrôle d'un individu et est totalement indépendant des stratégies, événements, processus ou individus. Le deuxième niveau, les tâches de contrôle, porte sur les buts à atteindre sans égard à la manière de les atteindre ni par qui. Le troisième niveau, les stratégies, porte sur la manière d'atteindre les buts. Le quatrième niveau, socio-organisationnel, porte sur les relations entre les différents acteurs impliqués dans l'application des stratégies pour atteindre les buts. Le cinquième niveau, les compétences, porte sur l'analyse des individus.

Normalement, l'analyse menant à la conception d'une IE devrait inclure les cinq niveaux identifiés à la figure 4.1. Toutefois, tel que mentionné précédemment, la force d'une IE est attribuée principalement à la représentation du domaine de travail sous la forme d'une hiérarchie fonctionnelle. De plus, la littérature fait état de très peu d'IE intégrant d'autres niveaux que le domaine de travail. C'est pour cette raison que ce chapitre s'intéresse uniquement à ce niveau d'analyse.

Une analogie s'impose afin de mieux comprendre la notion de domaine de travail: le domaine de travail est aux stratégies ce que les cartes routières sont aux directions. En effet, quelle que soit la source de perturbations du domaine de travail (ex. : une rue bloquée), il est possible de trouver un nouvel ensemble de directions (stratégies) pour atteindre la destination (le but fixé) si la carte routière (représentation du domaine de travail) constitue la principale source d'informations. Toutefois, si la principale source d'informations est composée de directions (stratégies), celles-ci peuvent devenir caduques devant certains événements imprévus (ex. : un accident) entraînant une perturbation du domaine de travail (ex. : une rue bloquée). Par conséquent, le domaine de travail impose un ensemble de contraintes qui exclut les actions qui ne peuvent être réalisées pour atteindre un but.

4.2. Qu'est-ce qu'une hiérarchie d'abstraction et de décomposition?

La HAD ne se préoccupe pas des tâches ou des outils, mais seulement des buts et informations (variables et liens) nécessaires à la compréhension de la dynamique du domaine de travail. Elle consiste à structurer les informations portant autant sur les aspects physiques que fonctionnels du domaine de travail, et ce, à travers différents niveaux d'abstraction et de décomposition. La figure 4.2 illustre le cadre générique de la HAD et la figure 4.3 illustre une application de la HAD au contrôle d'un processus thermo-hydraulique. Cette section présente une recension de la littérature sur la HAD divisée par thèmes : abstraction, décomposition, relations.

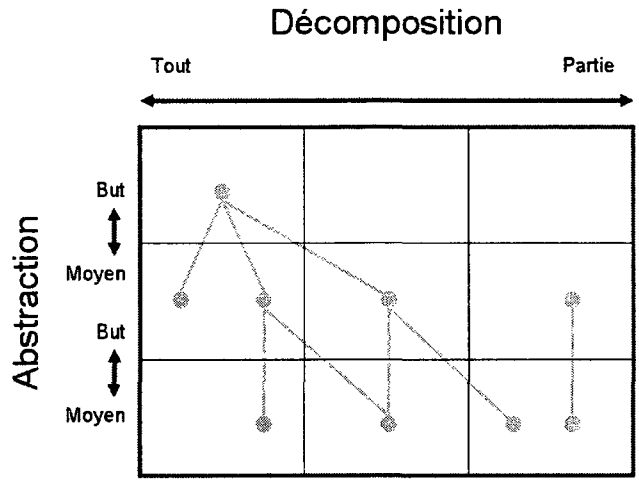


Figure 4.2 – Cadre générique d'une hiérarchie d'abstraction et de décomposition.

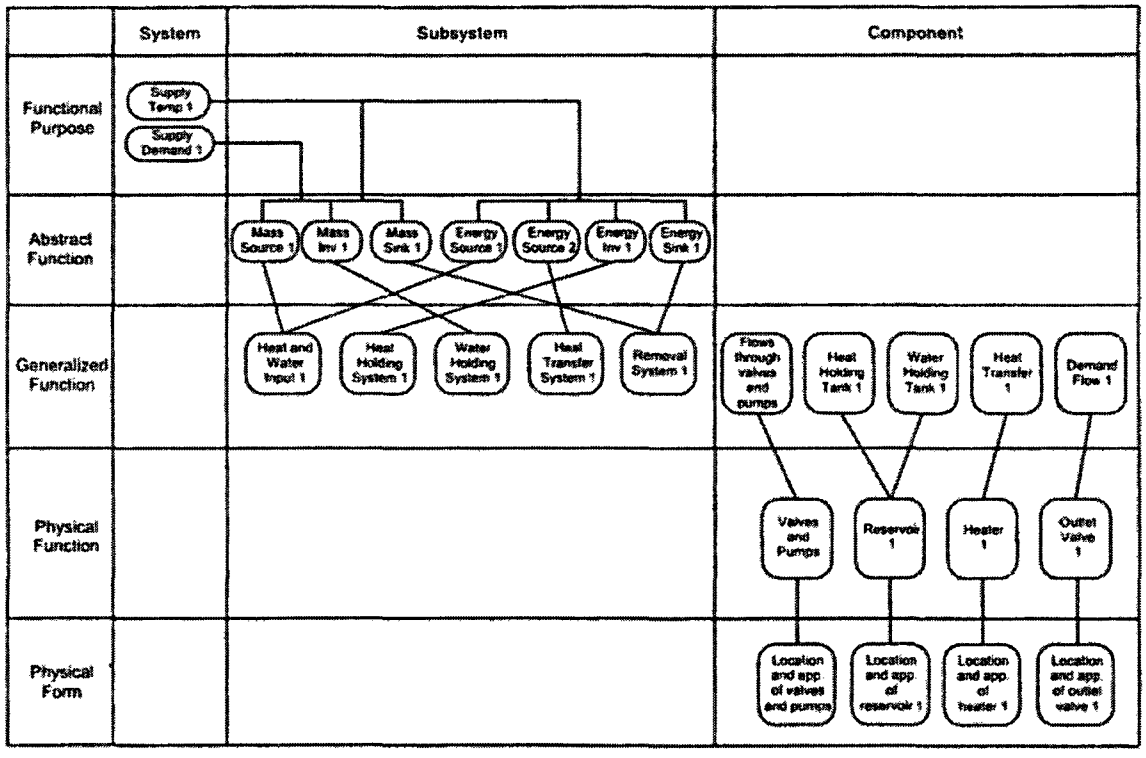


Figure 4.3 – Application de la hiérarchie d'abstraction et de décomposition au contrôle d'un processus thermo-hydraulique (Vicente 1999a, p. 175).

4.2.1. Nœud

Chaque case d'une HAD contient des nœuds. Un nœud représente un objet, une fonction ou une valeur du domaine de travail qui correspond à une intersection particulière de niveau d'abstraction et de niveau de décomposition (Skilton, Cameron et Sanderson 1998). Par exemple, à la figure 4.3, la masse en inventaire (*Mass Source 1*), l'énergie en entrée (*Energy Inv 1*), le transfert de chaleur (*Heat Transfer 1*), le réservoir (*Reservoir 1*) sont tous des nœuds.

4.2.2. Abstraction

Un niveau d'abstraction regroupe des nœuds qui portent sur une perspective particulière du domaine de travail. Tel qu'illustré à la figure 4.2, les nœuds des différents niveaux d'abstraction s'appuient sur des relations « buts-moyens », ce qui signifie que pour toute paire verticale de niveau d'abstraction, l'élément du niveau supérieur décrit le but à atteindre (le « pourquoi? ») et les éléments associés du niveau inférieur décrivent les moyens utilisés (le « comment? ») pour atteindre ce but. De cette façon, le niveau d'abstraction le plus élevé décrit les buts du système, c'est-à-dire sa raison d'être, et le niveau d'abstraction le plus bas décrit généralement les aspects physiques de celui-ci, c'est-à-dire ceux qui peuvent être manipulés ou qui sont perceptibles dans l'environnement. Les niveaux d'abstraction intermédiaires portent sur les aspects fonctionnels entre ces deux niveaux d'abstraction. En d'autres mots, le passage d'un niveau d'abstraction à un autre implique un changement dans les concepts et la structure de représentation ainsi que dans les informations utilisées pour caractériser le système à ce niveau d'abstraction (Rasmussen 1983).

À la suite d'interventions dans le domaine du contrôle de processus industriels, Rasmussen (1983) a défini la structure de la HAD comme ayant les cinq niveaux d'abstraction suivants, du plus élevé au plus bas : buts fonctionnels, fonctions abstraites,

fonctions généralisées, fonctions physiques et forme physique. Toutefois, Rasmussen et Vicente (1992) mentionnent que le nombre de niveaux et leur nom peuvent varier d'un domaine à l'autre.

La HAD de la figure 4.3 s'appuie sur ces cinq niveaux d'abstraction. La raison d'être du système est définie au niveau supérieur. Pour ce système, les buts fonctionnels sont (a) de préserver une température spécifique de l'eau dans chaque réservoir et (b) de préserver assez d'eau dans chaque réservoir pour répondre à la demande externe. Pour ce faire, le domaine de travail est constitué de pompes, de valves, de réservoirs et de brûleurs. Tous ces éléments sont identifiés au niveau d'abstraction inférieur représentant la forme physique de ces éléments.

4.2.3. Décomposition

Quant à elle, la dimension de décomposition permet de scinder le système en sous-ensembles de composants qui sont responsables du bon fonctionnement de celui-ci. En somme, elle sert à décomposer le tout en ses parties. Par conséquent, un élément représenté à une colonne particulière est composé d'éléments représentés à la colonne immédiatement à sa droite. Inversement, un élément représenté à une colonne particulière est une partie d'un élément représenté à la colonne immédiatement à sa gauche. Par exemple, Hajdukiewicz, Vicente, Doyle, Milgram et Burns (2001) décomposent le corps humain en système, organe, tissu et cellule; chacun de ces cinq éléments correspond à une colonne dans cet ordre. La figure 4.4 illustre cette décomposition. Cette HAD permet d'identifier le champ de compétence de l'anesthésiste et celui du chirurgien.

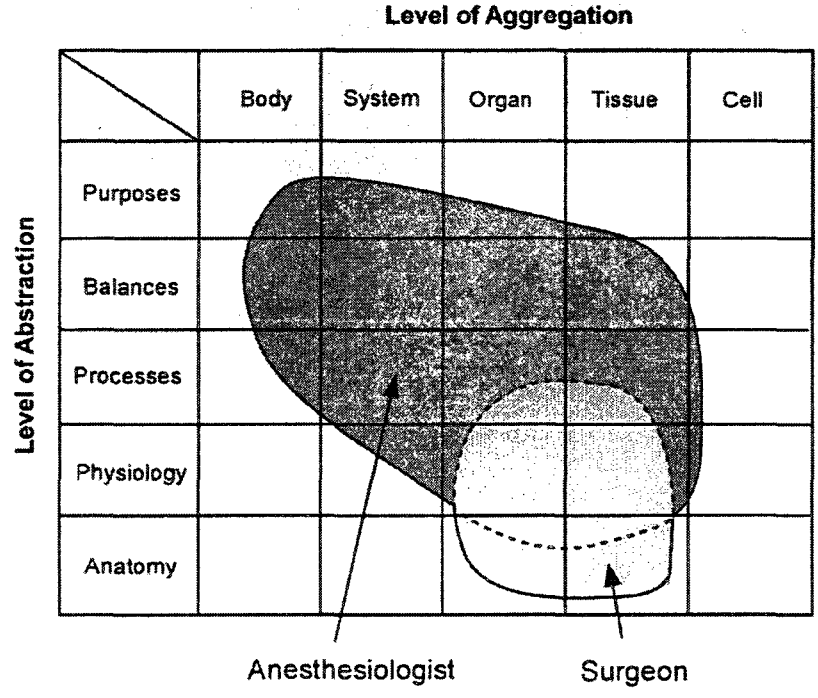


Figure 4.4 – Hiérarchie d'abstraction et de décomposition pour le corps humain (Hajdukiewicz, Vicente, Doyle, Milgram et Burns 2001).

4.2.4. Relations

Les relations entre les nœuds peuvent être de différents types. Bisantz et Vicente (1994), Skilton, Cameron et Sanderson (1998) ainsi que Burns et Hajdukiewicz (2004, p. 27) présentent trois types de relations : causale, fonctionnelle et topologique.

4.2.4.1. Relation causale

La relation causale représente une relation « tout-parties ». Les deux nœuds sont au même niveau d'abstraction, mais appartiennent à des niveaux de décomposition contigus. Cette relation entre deux nœuds indique que le nœud de l'extrémité gauche représente la partie du nœud de l'extrémité droite. Inversement, le nœud de l'extrémité droite représente la partie du nœud de l'extrémité gauche.

4.2.4.2. Relation fonctionnelle

La relation fonctionnelle représente une relation « but-moyens ». Les deux nœuds appartiennent à des niveaux d'abstraction contigus, qu'ils appartiennent ou non au même niveau de décomposition. Cette relation entre deux nœuds indique que le nœud de l'extrémité supérieure représente le but recherché par le nœud de l'extrémité inférieure. Inversement, le nœud de l'extrémité inférieure représente le moyen pour atteindre le but représenté par le nœud de l'extrémité supérieure. Le réseau de relations fonctionnelles d'une HAD constitue la hiérarchie fonctionnelle du domaine de travail.

4.2.4.3. Relation topologique

La relation topologique peut représenter soit une proximité spatiale entre deux nœuds ou une connexion physique entre eux dépendamment du niveau d'abstraction (Bisantz et Vicente 1994). Les deux nœuds sont au même niveau d'abstraction et au même niveau de décomposition.

4.3. Problèmes avec la hiérarchie d'abstraction et de décomposition

La littérature sur la HAD fait état de deux problèmes majeurs qui semblent bloquer son utilisation au sein de la communauté de praticiens. Cette section présente une description approfondie de chacun d'eux. Le premier problème porte sur la HAD en elle-même et le deuxième problème porte sur l'utilisation de la HAD en CIE.

4.3.1. La sémantique de la de hiérarchie d'abstraction et de décomposition est imprécise

Selon Lind (1999, 2003), le seul guide permettant comprendre et de créer une HAD est constitué des quelques exemples incomplets présentés dans littérature qui portent sur des domaines très particuliers. De plus, ces exemples présentent différentes manières

d'appliquer la HAD, ajoutant ainsi de la confusion à la sémantique de cette technique de représentation de domaines de travail. Cette section décrit la nature de ce problème en portant, en premier lieu, sur la dimension d'abstraction et, en second lieu, sur la dimension de décomposition.

4.3.1.1. Dimension d'abstraction

L'objectif de scinder la représentation d'un domaine de travail particulier en différents niveaux d'abstraction est d'en dégager une hiérarchie fonctionnelle. Tel que mentionné précédemment, celle-ci est une structure mettant en relation, à différents niveaux d'abstraction, les éléments qui composent un domaine de travail particulier. Le niveau d'abstraction le plus élevé est associé aux buts du système que représente le domaine de travail et le niveau le plus bas est associé à l'environnement pouvant être manipulé pour atteindre les buts. La hiérarchie fonctionnelle est obtenue en plaçant ces éléments dans les différentes cases de la HAD, c'est-à-dire aux intersections entre un niveau d'abstraction particulier et un niveau de décomposition particulier telle qu'illustrée aux figures 4.2 et 4.3.

Ham et Yoon (2001) apportent une justification expérimentale à la pertinence de la hiérarchie fonctionnelle. Dans une expérience, ils ont comparé différents prototypes d'IE. Certains comportaient des variables associées à des niveaux spécifiques d'une HAD, mais sans rendre explicites les relations entre celles-ci. D'autres présentaient les relations fonctionnelles entre les variables principalement par des regroupements. Ces derniers prototypes ont permis aux utilisateurs d'obtenir de meilleures performances notamment pour des tâches de diagnostic. Ham et Yoon (2001) concluent donc en spécifiant que pour tirer les bénéfices d'une IE, il est nécessaire, entre autres, de représenter la hiérarchie fonctionnelle, c'est-à-dire les relations « buts-moyens », entre

les variables; il ne s'agit donc pas de simplement représenter les informations appartenant aux différents niveaux d'abstraction.

Tel que mentionné précédemment, une HAD comporte généralement cinq niveaux d'abstraction. Ces derniers semblent convenir à la description de domaines de travail portant sur le contrôle de processus industriels. Toutefois, la littérature présente quelques cas pour lesquels le nombre de niveaux est autre que cinq et les noms de chaque niveau ont dû être modifiées (ex. : Hajdukiewicz, Vicente, Doyle, Milgram et Burns (2001), Achonu et Jamieson (2003), Dainoff, Dainoff, McFeeters (2004)). En somme, si les cinq niveaux d'abstraction habituellement employés ne conviennent pas au domaine de travail à modéliser, il importe de les redéfinir (nombre et nom) afin qu'ils soient représentatifs de celui-ci.

Est-ce que les informations affichées par l'IE doivent être regroupées par niveaux d'abstraction? Est-il nécessaire de définir des niveaux d'abstraction et de les nommer? Lind (1999, 2003) critique sévèrement les cinq niveaux d'abstraction définis par Rasmussen (1983). Sans entrer dans les détails de sa réflexion, il indique que la sémantique employée par la HAD dans sa forme actuelle nécessite une profonde révision. En effet, sans mettre en doute l'importance de la hiérarchie fonctionnelle, il indique que la définition de ces cinq niveaux d'abstraction ajoute de la confusion quant à la modélisation de la structure d'un domaine de travail particulier; il est souvent difficile de savoir à quel niveau d'abstraction est associé un élément particulier.

L'argumentation de Lind (1999, 2003) repose sur son expérience personnelle avec la HAD. Il existe par contre une justification expérimentale qui abonde dans ce sens. Burns (2000) a réalisée une expérience visant à présenter de trois manières les informations d'une HAD dans une IE. La première manière consiste à toutes les représenter dans la

même fenêtre. La deuxième manière consiste à présenter, dans la surface d'affichage, quatre fenêtres présentant chacune les informations sur un niveau d'abstraction spécifique. La troisième manière consiste à présenter individuellement, dans la surface d'affichage, les quatre fenêtres présentant chacune les informations sur un niveau d'abstraction spécifique. Les résultats ont démontré que le meilleur temps de diagnostic était attribué au prototype d'IE dont toutes les informations étaient présentes dans la même fenêtre. Burns (2000) attribue ce résultat, en citant Tufte (1983), non pas au simple fait que cette fenêtre comportait plus d'informations, mais au fait que les informations étaient présentées en relation avec d'autres. Par conséquent, il semble raisonnable de conclure que le découpage de la hiérarchie fonctionnelle en niveaux d'abstraction spécifiques ne semble pas avoir d'impact sur la performance de l'utilisateur. L'important est que les relations fonctionnelles entre les informations soient explicites.

4.3.1.2. Dimension de décomposition

En ce qui concerne la dimension de décomposition, celle-ci doit être définie pour chaque domaine de travail à modéliser. Tel que mentionné précédemment, la colonne d'extrême gauche représente le tout et les colonnes à sa droite représentent ses parties. Toutefois, cette règle ne semble pas être appliquée strictement. En effet, plusieurs exemple de HAD substituent la dimension de décomposition par une dimension de collaboration comme c'est le cas dans Rasmussen, Pjetersen et Goodstein (1994, p. 47) ainsi que Burns, Kuo et Ng (2003). Chaque colonne de cette dimension correspond à différentes entités du domaine de travail qui collaborent les unes avec les autres. Par exemple, la figure 4.5 illustre une HAD pour la gestion de réseaux informatiques. Les colonnes représentent des couches de la hiérarchie de communication. Les liens entre chacune des couches ne représentent pas des parties et des touts, mais sont plutôt de nature collaborative, c'est-à-dire que les éléments d'une couche interagissent avec les éléments d'une autre couche.

Même si Burns, Kuo et Ng (2003) considèrent que les relations horizontales entre les nœuds sont similaires à une relation « tout-parties », ceci est en contradiction avec la décomposition de la HAD des figures 4.2 et 4.3.

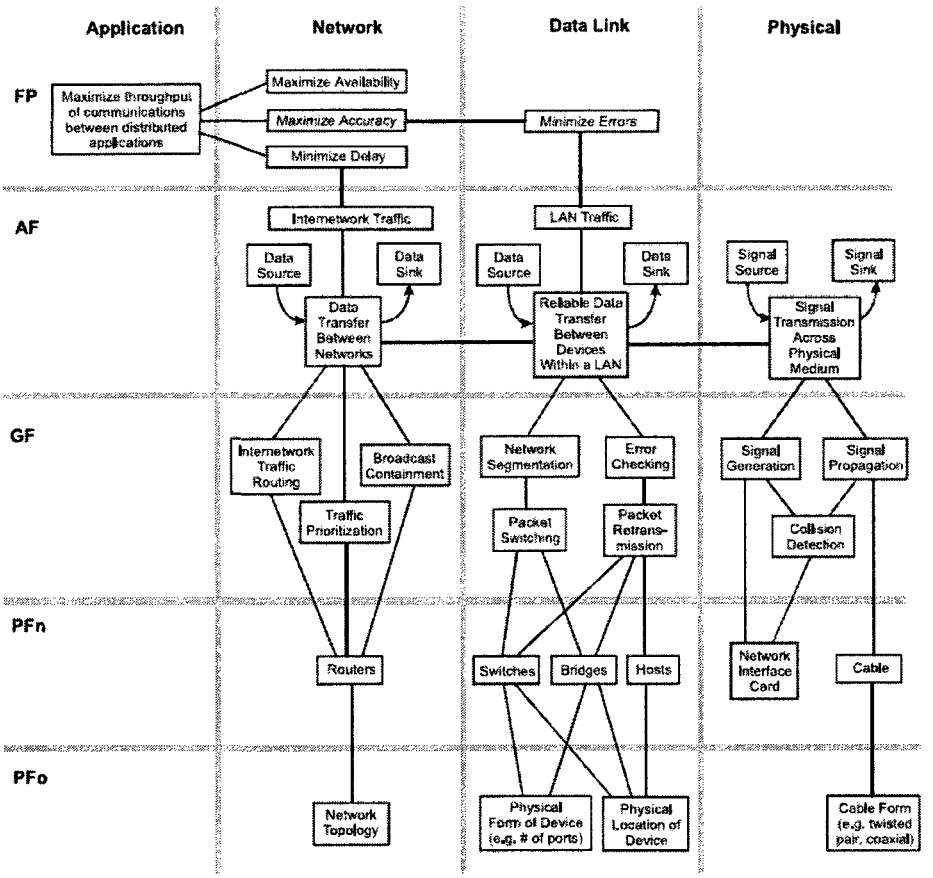


Figure 4.5 – Hiérarchie d'abstraction et de décomposition pour la gestion de réseaux informatiques (Burns, Kuo et Ng 2003).

Le réel problème est que cette différence dans la nature des relations entre les colonnes de la HAD n'est pas visuellement explicite. Par conséquent, une personne qui ne connaît pas le domaine de travail modélisé, en l'occurrence un réseau informatique, peut facilement confondre des relations de nature « tout-parties » avec des relations de nature

collaborative. Il importe donc de distinguer visuellement la nature des relations entre les colonnes de la dimension de décomposition de la HAD.

4.3.2. La conception d'interfaces écologiques nécessite plus qu'une hiérarchie d'abstraction et de décomposition

Vicente (2002) affirme que la transformation d'une HAD en une IE exige une grande part de créativité. En effet, la tâche de concevoir la forme visuelle des éléments de la HAD comporte plusieurs degrés de liberté. Burns et Hajdukiewicz (2004) présentent un ouvrage portant sur le sujet. Toutefois, implicitement selon les auteurs, cette transformation nécessite plus de détails que ce qui est présenté par les éléments de la HAD. Ces auteurs indiquent que deux étapes sont nécessaires pour permettre l'utilisation de la HAD, une fois celle-ci élaborée, dans la conception d'une IE[17].

La première étape consiste à dresser une liste des variables à partir de la HAD. Comme il est indispensable de pouvoir afficher l'état des différents objets du domaine de travail, il est nécessaire d'aller vers un niveau de détail plus fin que l'identification de concepts représentés par des nœuds tels qu'illustrés aux figures 4.2 et 4.3. L'état d'un objet est caractérisé par la valeur des attributs qui le compose. Puisque ces attributs doivent être reliés ensemble sous la forme d'équations mathématiques afin de constituer la structure du domaine de travail, ils sont considérés comme étant des termes (variables ou constantes).

La deuxième étape consiste en la description de la nature des relations entre ces termes. À ce sujet, Lind (1999) affirme que de représenter les relations entre les concepts par des lignes les reliant les uns aux autres comme c'est le cas avec la HAD contribue à une sémantique vague. Afin de préciser la nature des relations, une liste d'équations,

[17] Voir le chapitre 4 de Burns et Hajdukiewicz (2004).

représentant la structure du domaine de travail, doit généralement être dressée à partir de la liste des variables selon Burns et Hajdukiewicz (2004). Tel que mentionné précédemment, les expressions logiques représentent les contraintes du domaine de travail.

Le tableau 4.1 présente différents exemples de contraintes multi-variées présentées par Burns et Hajdukiewicz (2004, p. 94). Pour une expression utilisant un opérateur d'égalité, le résultat du côté droit de l'expression doit être le même que le résultat du côté gauche de l'expression. Dans le cas contraire, il y a un écart et donc une anomalie. Les termes utilisés dans ces expressions sont associés à différents niveaux d'abstraction. La structure qui relie ces termes à différents niveaux d'abstraction constitue la hiérarchie fonctionnelle. Les contraintes, termes et hiérarchie fonctionnelle sont des éléments indispensables pour la conception de l'IE, mais ne sont pas explicites dans la HAD.

Tableau 4.1 – Exemples de contraintes multi-variées par niveau d'abstraction [adapté et traduit de Burns et Hajdukiewicz (2004, p. 94)].

Équation	*Fonction abstraite*	*Fonction généralisée*	*Fonction physique*	*Forme physique*
Transfert thermique $Q = m * Cp * dT$	m, Q	dT	Cp	
Quantité de mouvement $mu = mv$	Mu, m	v		
Force $F = ma$	F, m	a		
Couple = F * d	F	d		
Loi des gaz parfaits $pV = nRT$		p, V, T	R	n
Décroissance nucléaire $N = N_0e^{-(0,693t/T_{\frac{1}{2}})}$	N, N_0	t	$T_{\frac{1}{2}}$	
$E = mc^2$	E, m		C, medium	

En somme, la HAD ne permet pas de représenter la structure du domaine de travail à un niveau de détail assez précis pour la conception d'IE. Afin de palier ce problème, il est nécessaire d'élaborer une liste de variables et une liste d'équations. D'après Burns et Hajdukiewicz (2004), la HAD et ces deux listes sont essentielles à la conception d'IE.

4.4. Méta-modèle de la hiérarchie d'abstraction et de décomposition

Cette section présente le méta-modèle de la HAD. Un méta-modèle de la HAD consiste à mettre en relation d'une manière rigoureuse tous les éléments manipulés par l'analyste de domaines de travail qui sont identifiés à la section précédente. Les éléments principaux du méta-modèle sont les suivants : nœud, niveau d'abstraction, niveau de décomposition et relation.

Le méta-modèle est scindé en deux parties. La première partie, illustrée à la figure 4.5 sous forme d'un diagramme de classes du langage UML (OMG 2008b), présente la structure de base. Toutefois, le diagramme de classes ne permet pas de représenter toutes les règles structurelles. La deuxième partie, illustrée à la figure 4.6 en langage OCL (OMG 2006), permet de préciser celles-ci.

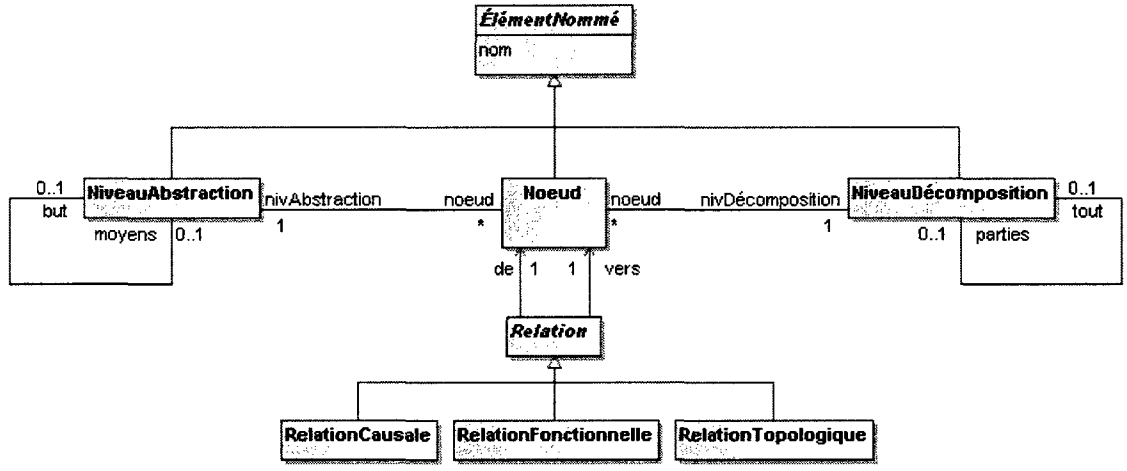


Figure 4.5 – Méta-modèle de la hiérarchie d'abstraction et de décomposition.

La figure 4.7 illustre les instances du méta-modèle de la HAD pour un sous-ensemble de la HAD du processus thermo-hydraulique de la figure 4.3. Ce sous-ensemble comprend les cinq niveaux d'abstraction, les trois niveaux de décomposition, trois nœuds et deux relations fonctionnelles. Les règles d'association définies à la figure 4.6 sont observées. Par exemple, selon la règle 1, aucun niveau d'abstraction n'est associé au même niveau d'abstraction pour représenter à la fois son but et ses moyens; il s'agit toujours de deux niveaux d'abstraction différents. Également, selon la règle 5, les deux relations fonctionnelles sont associées à des nœuds qui sont associés à des niveaux d'abstraction contigus.

```
-- Règle 1 :
-- Pour un niveau d'abstraction, le niveau qui représente son "but"
-- ne peut en même temps représenter ses "moyens" et vice versa
context NiveauAbstraction
inv: but <> moyens

-- Règle 2 :
-- Pour un niveau de décomposition, le niveau qui représente son "tout"
-- ne peut en même temps représenter ses "parties" et vice versa
context NiveauDécomposition
inv: self.tout <> self.parties

-- Règle 3 :
-- Les deux extrémités d'une relation ne peuvent pas
-- être associées au même nœud
context Relation
inv: de <> vers

-- Règle 4 :
-- Les deux nœuds impliqués dans une relation causale
-- doivent être associés au même niveau d'abstraction
-- et à deux niveaux de décomposition contigus
context RelationCausale
inv: de.nivAbstraction = vers.nivAbstraction and
    (de.nivDécomposition = vers.nivDécomposition.tout or
     de.nivDécomposition = vers.nivDécomposition.parties)

-- Règle 5 :
-- Les deux nœuds impliqués dans une relation fonctionnelle
-- doivent être associés à des niveaux d'abstraction contigus
context RelationFonctionnelle
inv: de.nivAbstraction = vers.nivAbstraction.but or
     de.nivAbstraction = vers.nivAbstraction.moyens

-- Règle 6 :
-- Les deux nœuds impliqués dans une relation topologique
-- doivent être associés au même niveau d'abstraction
-- et au même niveau de décomposition
context RelationTopologique
inv: de.nivAbstraction = vers.nivAbstraction and
     de.nivDécomposition = vers.nivDécomposition
```

Figure 4.6 – Règles d'association du méta-modèle de la hiérarchie d'abstraction et de décomposition.

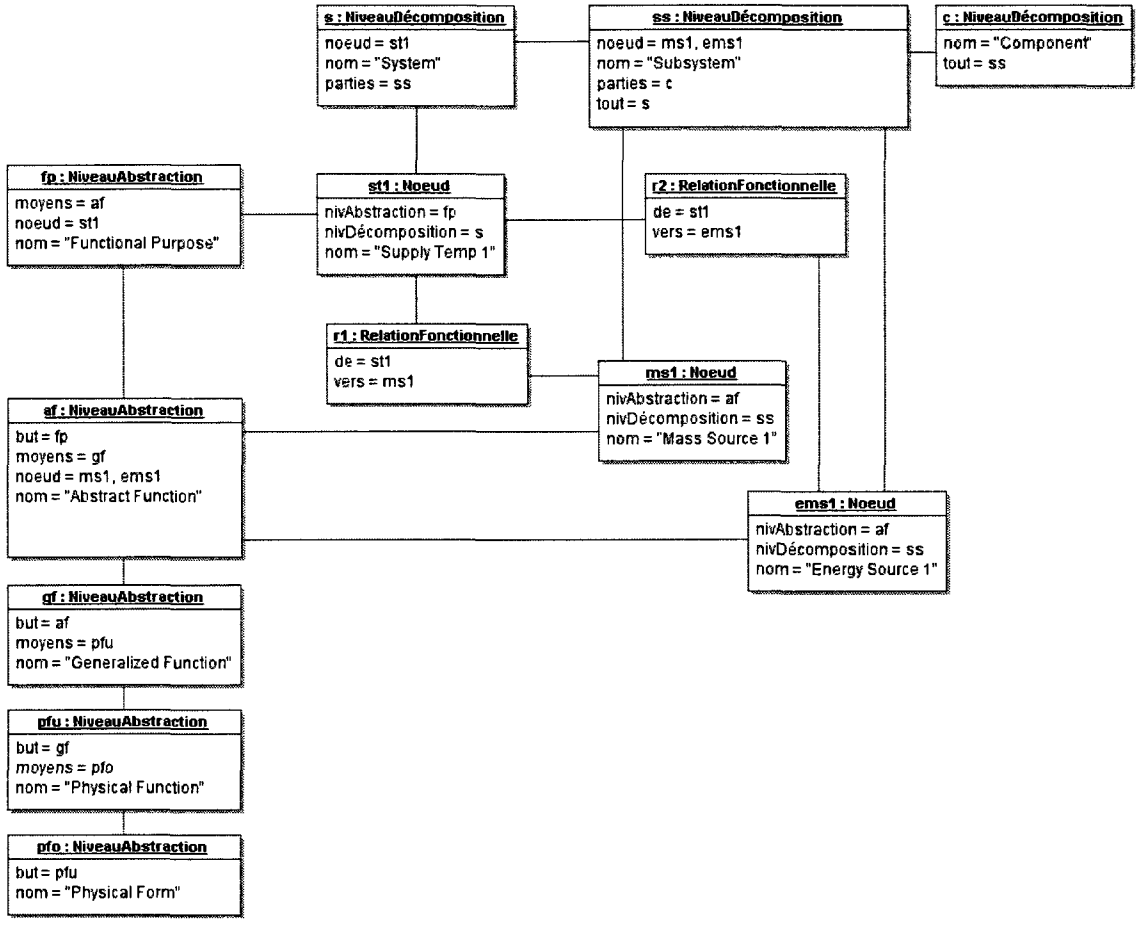


Figure 4.7 – Instances du méta-modèle de la hiérarchie d'abstraction et de décomposition pour un sous-ensemble de la hiérarchie d'abstraction et de décomposition du processus thermo-hydraulique.

Le méta-modèle de la HAD présenté dans cette section permet d'aller plus loin que les exemples présentés dans la littérature pour comprendre cette technique de représentation de domaines de travail. Entre autres, la hiérarchie fonctionnelle peut être observée à travers les relations fonctionnelles entre les nœuds de différents niveaux d'abstraction. De plus, ce méta-modèle permet de définir les cinq niveaux d'abstraction généralement employés, mais il permet également de définir un nouvel ensemble. Quoi qu'il en soit, le méta-modèle spécifie les associations entre ces niveaux et les nœuds. Bien que cela solutionne une partie du problème décrit à la section 4.3.1, il n'en demeure pas moins

qu'il existe toujours une difficulté méthodologique reliée à la manière de définir des niveaux d'abstraction.

En ce qui concerne les niveaux de décomposition, le méta-modèle spécifie la manière dont ceux-ci sont associés les uns aux autres ainsi qu'aux nœuds. Toutefois, leur interprétation est limitée à des relations « tout-parties »; aucun élément du méta-modèle ne tient compte des relations de collaborations. La raison pour ne pas avoir ajouté une dimension de collaboration est que, tel que mentionné précédemment, Burns, Kuo et Ng (2003) considèrent que les relations horizontales entre les nœuds sont de nature « tout-parties ». En somme, puisqu'aucun élément n'a été spécifié à cette fin dans la syntaxe abstraite, il n'est pas possible de distinguer ces éléments dans la syntaxe concrète.

4.5. Évaluation de la compatibilité de la hiérarchie d'abstraction et de décomposition

Cette section vise à évaluer la compatibilité entre le méta-modèle de la HAD et le méta-modèle d'une IE décrit à la section 3.3. En d'autres mots, il s'agit de vérifier si la HAD permet de représenter tous les éléments nécessaires à la CIE.

La section 4.3.2 décrit le problème associé à la transformation d'une HAD en une IE. Il est indiqué que cette transformation ne peut être réalisée à partir d'une HAD uniquement; il est nécessaire de recourir à une liste de variables et à une liste d'équations. L'évaluation est donc scindée en deux parties. La première consiste à évaluer la compatibilité de la HAD sans la liste des variables ni la liste des équations. La deuxième partie consiste à évaluer la compatibilité de la HAD avec ces dernières.

4.5.1. Méthode d'évaluation de la compatibilité

La transformation d'une HAD en une IE nécessite au préalable d'établir des correspondances entre les éléments du méta-modèle de la HAD et les éléments du méta-modèle d'une IE. Si chaque élément du méta-modèle de la HAD correspond à un élément du méta-modèle d'une IE, il y a une compatibilité parfaite entre les méta-modèles. Dans le cas contraire, c'est signe qu'une notation secondaire est nécessaire pour compenser. L'évaluation de la compatibilité consiste donc à analyser les correspondances entre ces éléments.

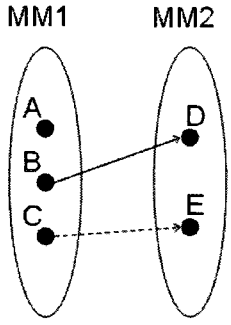
Chaque analyse de correspondance se fait en deux étapes. La première étape consiste à évaluer la correspondance du méta-modèle de la HAD vers le méta-modèle d'une IE. La deuxième étape consiste à évaluer la correspondance du méta-modèle d'une IE vers le méta-modèle de la HAD.

Pour chaque élément, une note de 2 signifie une correspondance parfaite, une note de 1 signifie une correspondance partielle et une note de 0 signifie aucune correspondance. Les notes sont cumulées et comparées au résultat maximal. Ce dernier est 2 (la note maximale) multiplié par le nombre d'éléments du méta-modèle. Le résultat de la division entre ces deux termes (notes cumulées et résultat maximal) correspond au taux de correspondance, affiché en pourcentage.

La figure 4.8 présente un exemple d'analyse de correspondance pour les éléments de deux méta-modèles nommés MM1 et MM2. Dans le premier cas, de MM1 vers MM2, le taux de correspondance est de 50%. Le résultat maximal est de 6 et la note cumulée est de 3. L'élément A n'a aucune correspondance, l'élément B a une correspondance parfaite et l'élément C a une correspondance partielle. Par contre, dans le deuxième cas, de MM2 vers MM1, le taux de correspondance est de 75%. Le résultat maximal est de 4

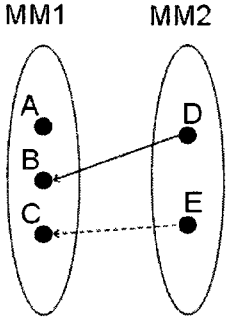
et la note cumulée est de 3. L'élément D a une correspondance parfaite et l'élément E a une correspondance partielle.

(a)



MM1	MM2	Note
A	-	0
B	D	2
C	E	1
		3 / 6 (50%)

(b)



MM2	MM1	Note
D	B	2
E	C	1
		3 / 4 (75%)

Figure 4.8 – Exemple d'évaluation de la correspondance (a) de MM1 vers MM2 et (b) de MM2 vers MM1.

Trois constats peuvent être tirés de cet exemple. Le premier constat est que l'élément A du méta-modèle MM1 est superflu puisque le méta-modèle MM2 n'a aucun élément qui lui correspond. Le deuxième constat est que l'élément B du méta-modèle MM1 correspond parfaitement à l'élément D du méta-modèle MM2. Par conséquent, chaque élément de type B peut être directement transformé en un élément de type D. Le troisième constat est que l'élément C du méta-modèle MM1 correspond partiellement à l'élément E du méta-modèle MM2 ce qui requiert une intervention humaine pour transformer un élément de type C en un élément de type E.

4.5.2. Évaluation de la compatibilité sans la liste des variables ni la liste des équations

Puisque le méta-modèle de la HAD contient six éléments concrets[18], le résultat maximal est de 12. Le tableau 4.2 présente les résultats de cette analyse.

Tableau 4.2 – Correspondance des éléments du méta-modèle de la hiérarchie d'abstraction et de décomposition vers les éléments du méta-modèle d'une interface écologique.

HAD	IE	Note
Noeud	Concept	2
Niveau d'abstraction	-	0
Niveau de décomposition	Concept (inclusion)	2
Relation causale	Opérateur	1
Relation fonctionnelle	Opérateur	1
Relation topologique	-	0
		6 / 12 (50%)

Les deux seuls éléments du méta-modèle de la HAD qui ont une correspondance parfaite sont le nœud et le niveau de décomposition. Le nœud correspond à un concept du méta-modèle de l'IE. Quant au niveau de décomposition, il correspond également au concept puisque ce dernier peut inclure d'autres concepts. Cette inclusion indique une relation « tout-parties ». Par exemple, la figure 3.7 (b) illustre un système qui se décompose en deux sous-systèmes[19] qui incluent des composants comme des pompes, des réservoirs et des brûleurs. Ces trois niveaux de décomposition sont représentés par les trois colonnes de la HAD de la figure 4.3.

[18] Les méta-classes *ÉlémentNommé* et `Relation` sont abstraites.

[19] Chaque sous-système correspond à un circuit de circulation d'eau qui débute à l'embranchement de gauche. Le premier sous-système, appelé A, est représenté par la portion du haut de la figure incluant la pompe PA et se terminant avec le réservoir 1. Le deuxième sous-système, appelé B, est représenté par la partie du bas de la figure incluant la pompe PB et se terminant avec le réservoir 2.

La relation causale et la relation fonctionnelle sont des éléments du méta-modèle de la HAD qui correspondent partiellement aux opérateurs du méta-modèle d'une IE. Selon ce dernier, les opérateurs indiquent une relation entre des termes. Quant aux relations causale et fonctionnelle, elles représentent des relations entre des nœuds. Or, les nœuds correspondent à des concepts qui regroupent des termes selon le méta-modèle d'une IE. En somme, la HAD ne va pas à un niveau de granularité aussi fin que requiert une IE. En effet, le niveau de granularité le plus fin de la HAD est le nœud. Quant à une IE, son niveau de granularité le plus fin est le terme. Chaque terme appartient à un concept qui correspond à un nœud de la HAD.

Il n'existe aucun élément du méta-modèle d'une IE qui correspond au niveau d'abstraction et à la relation topologique. En ce qui concerne le premier, il s'agit d'une situation surprenante à première vue puisque le scindement d'un domaine de travail par niveaux d'abstraction est le pilier de la philosophie de la HAD. Néanmoins, cette situation ne semble pas si surprenante en considérant les résultats d'expériences mentionnés à la section 4.3.1.1. Ces derniers indiquent que le regroupement par niveau d'abstraction des informations affichées par une IE n'améliore en rien la performance de l'utilisateur. L'important est que la hiérarchie fonctionnelle soit représentée.

Bien que la hiérarchie fonctionnelle représente la hiérarchie de « but-moyens » autant pour la HAD que pour une IE, elle s'appuie sur un ensemble différent d'éléments. Pour la HAD, elle est constituée du réseau de relations fonctionnelles. Pour une IE, il s'agit du réseau d'opérateurs mathématiques représenté par des agencements de formes géométriques. Encore une fois, le problème de granularité suffisante se manifeste.

Cette évaluation soutient l'hypothèse que la raison d'être des niveaux d'abstraction n'est autre que pour soutenir la définition de la hiérarchie fonctionnelle telle que mentionnée à la section 4.3.1.1. Cette dernière est obtenue après avoir (1) défini un ensemble de niveaux d'abstraction, (2) assigné des nœuds à chacun de ces niveaux et (3) relié ces nœuds par des relations fonctionnelles. Avec une HAD, la hiérarchie fonctionnelle ne peut exister si ce n'est que suite à ces trois étapes.

Le taux de correspondance est de 50% pour cette analyse entre le méta-modèle de la HAD et le méta-modèle d'une IE. Toutefois, qu'en est-il de la situation inverse? Obtiendrait-on le même résultat entre le méta-modèle d'une IE et le méta-modèle de la HAD? Le tableau 4.3 présente les résultats de cette analyse. Puisque le méta-modèle d'une IE contient trois éléments[20], le résultat maximal est 6.

Tableau 4.3 – Correspondance des éléments du méta-modèle d'une interface écologique vers les éléments du méta-modèle de la hiérarchie d'abstraction et de décomposition.

IE	HAD	Note
Concept	Nœud	2
Terme	-	0
Opérateur	Relation causale Relation fonctionnelle	1
		3 / 6 (50%)

Tel qu'observé à l'analyse précédente, le concept correspond parfaitement au nœud et l'opérateur correspond partiellement à la relation causale et à la relation fonctionnelle. Le terme n'a toutefois aucune correspondance avec un élément du méta-modèle de la

[20] La méta-classe abstraite `Opérateur` a été retenue pour des raisons de simplicité. De plus, puisque la HAD ne distingue pas les constantes des variables, la méta-classe abstraite `Terme` a été retenue.

HAD. Avec 50%, le taux de correspondance de cette analyse n'est donc pas mieux que celui de l'analyse précédente.

4.5.3. Évaluation de la compatibilité avec la liste des variables et la liste des équations

Avec une si faible correspondance, comment est-il possible de concevoir une IE à partir d'une HAD? Tel que mentionné à la section 4.3.2, la HAD doit être accompagnée d'une liste de variables et d'une liste d'équations. Par exemple, la figure 4.9 présente la liste des variables et la liste des équations qui accompagne la HAD de la figure 4.3. Les constantes sont incluses dans la liste des équations.

Qu'arrive-t-il aux résultats de l'analyse de correspondance si les éléments de ces listes sont ajoutés? Pour ce faire, il faut au préalable identifier les éléments de ces listes. La liste des équations est en fait une liste d'expressions mathématiques où chaque expression logique correspond à une contrainte du domaine de travail. Par conséquent, les éléments de cette liste peuvent être représentés sous forme d'un méta-modèle des expressions mathématiques. La figure 4.10 illustre une partie de ce méta-modèle[21]. Quant aux éléments de la liste des variables, ils se retrouvent également dans ce méta-modèle puisque chaque variable identifiée dans cette liste est utilisée par une ou plusieurs expressions mathématiques.

[21] Plusieurs opérations, telles « plus petit ou égal » et les exposants, ont été omises pour des raisons de simplicité.

(a)

Temperature variables
- T1—Temperature of res. 1.
- T2—Temperature of res. 2

Mass variables
- D1—Demand (output) flowrate for res. 1
- D2—Demand (output) flowrate for res. 2
- MI1—Mass Input flowrate for res. 1
- MI2—Mass Input flowrate for res. 2
- V1—Volume of res. 1
- V2—Volume of res. 2

Energy variables
- E1—total Energy stored in res. 1
- E2—total Energy stored in res. 2
- EI1—Energy Input flowrate for res. 1
- EI2—Energy Input flowrate for res. 2
- EO1—Energy Output flowrate for res. 1
- EO2—Energy Output flowrate for res. 2

Heat transfer rates
- FH1—Flow from heater HTR1
- FH2—Flow from heater HTR2

Flowrates
- FA1—Flowrate from valve VA1
- FB1—Flowrate from valve VB1
- FA2—Flowrate from valve VA2
- FB2—Flowrate from valve VB2
- FPA—Flowrate from pump PA
- FPB—Flowrate from pump PB
- FVA—Flowrate from valve VA
- FVB—Flowrate from valve VB

Heater settings
- HTR1—setting for Heater of res. 1
- HTR2—setting for Heater of res. 2

Pump settings
- PA—setting of Pump in fws A
- PB—setting of Pump in fws B

Valve settings
- VA—setting of initial Valve in fws A
- VB—setting of initial Valve in fws B
- VA1—setting of Valve 1 in fws A
- VB1—setting of Valve 1 in fws B
- VA2—setting of Valve 2 in fws A
- VB2—setting of Valve 2 in fws B

(b)

Goals
1. Temperature setpoint
2. Demand satisfaction

Algebraic equations
1. $E1(t) = T1(t)V1(t)c_p\rho$ —relationship between energy, volume, and temperature
2. $EI1(t) = FH1(t) + c_pT_iMI1(t)$ —conservation of energy from heater and inflow
3. $MI1(t) = FA1(t) + FB1(t)$ —conservation of mass from two feedwater streams
4. $EO1(t) = D1(t)c_pT1(t)$ —energy leaving reservoir
5. $FA(t) = FA1(t) + FA2(t)$ —conservation of mass in feedwater stream
6. $FH1(t) = HTR1(t)$ —conservation of energy from heater
7. $FA1(t) = \dfrac{FA(t)VA1(t)}{VA1(t) + VA2(t)}$ —flow split relation
8. If pump is OFF then $FPA(t) = 0$,
   otherwise:
   IF $[VA1(t) + VA2(t)] > VA(t)$ THEN
   $FPA(t) = VA(t)$
   ELSE
   $FPA(t) = VA1(t) + VA2(t)$ —flow through pump
9. $FA(t) = FPA(t)$ —conservation of mass in pipe

State equations
1. $\dfrac{dT1(t)}{dt} = \dfrac{H1(t) - [FA1(t) + FB1(t)][T1(t) - T_i]cp}{V1(t)c_p\rho}$ —conservation of energy in reservoir
2. $\dfrac{dVI(t)}{dt} = \dfrac{FA1(t) + FB1(t) - D1(t)}{\rho}$ —conservation of mass in reservoir

Constants
- ρ—density of water
- c_p—specific heat capacity
- T_i—inlet water temperature

Figure 4.9 – (a) Liste des variables et (b) liste des équations pour le contrôle d'un processus thermo-hydraulique (Bisantz et Vicente 1994).

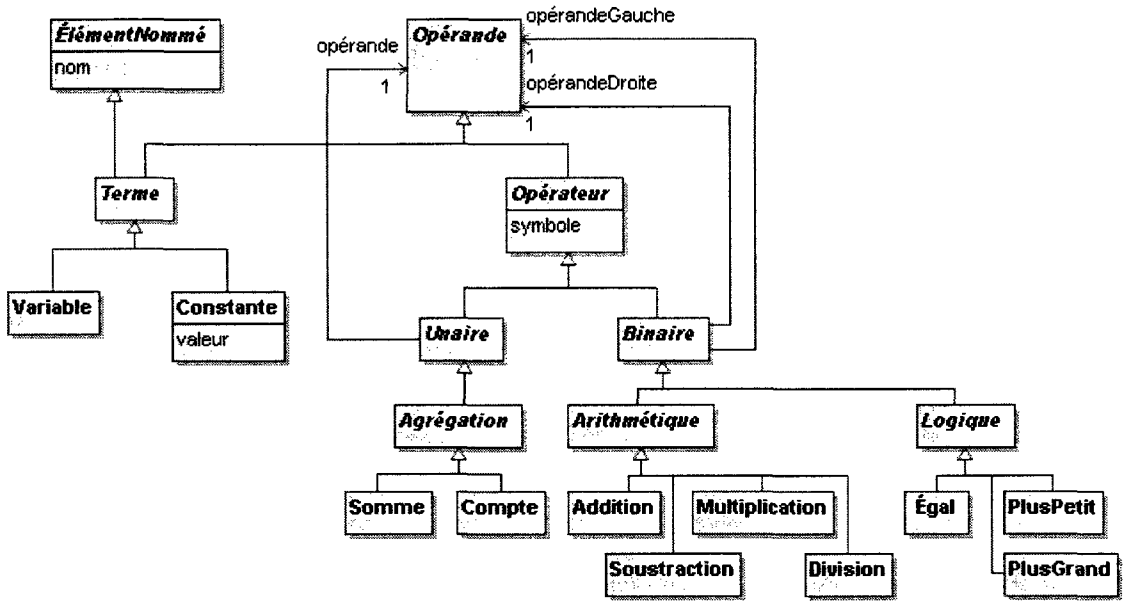


Figure 4.10 – Méta-modèle partiel d'une expression mathématique.

Le tableau 4.4 présente le résultat de l'analyse de correspondance entre les éléments du méta-modèle de la HAD, auxquels ont été ajoutés les éléments du méta-modèle des expressions mathématiques, et les éléments du méta-modèle d'une IE.

Tableau 4.4 – Correspondance des éléments du méta-modèle de la hiérarchie d'abstraction et de décomposition accompagnés des éléments du méta-modèle de expressions mathématiques vers les éléments du méta-modèle d'une interface écologique.

HAD	IE	Note
Noeud	Concept	2
Niveau d'abstraction	-	0
Niveau de décomposition	Concept (inclusion)	2
Relation causale	Opérateur	1
Relation fonctionnelle	Opérateur	1
Relation topologique	-	0
Terme	Terme	2
Opérateur	Opérateur	2
		10 / 16 (62,5%)

L'ajout des éléments du méta-modèle des expressions mathématiques aux éléments du méta-modèle de la HAD fait augmenter le taux de correspondance à 62,5%. Bien que le résultat soit plus élevé, la correspondance n'est toujours pas parfaite. Avant de tirer quelques conclusions que ce soit, encore une fois, il faut s'interroger sur la situation inverse.

Le tableau 4.5 présente les résultats de cette analyse. Le taux de correspondance est de 100%, ce qui veut dire qu'en ajoutant les éléments du méta-modèle des expressions mathématiques aux éléments du méta-modèle de la HAD, chaque élément du méta-modèle d'une IE a une correspondance parfaite.

Tableau 4.5 – Correspondance des éléments du méta-modèle d'une interface écologique vers les éléments du méta-modèle de la hiérarchie d'abstraction et de décomposition accompagnés des éléments du méta-modèle de expressions mathématiques.

IE	HAD	Note
Concept	Nœud	2
Terme	Constante Variable	2
Opérateur	Opérateur	2
		6 / 6 (100%)

Deux conclusions peuvent être tirées jusqu'à présent. La première conclusion est que ces résultats démontrent clairement l'insuffisance de la HAD en CIE; sans la liste des variables et la liste des équations, le concepteur d'IE ne possède pas toutes les informations nécessaires pour faire son travail.

La deuxième conclusion est que des éléments du méta-modèle de la HAD correspondent aucunement aux éléments du méta-modèle d'une IE. Par conséquent, l'analyste du domaine de travail, dont la tâche est de réaliser une HAD, tient compte de plus d'éléments que nécessaires pour le concepteur d'une IE. Tel que mentionné à la section 2.1.2, un modèle est une simplification. Ainsi, il doit représenter uniquement les éléments pertinents à la préoccupation actuelle, dans ce cas, la CIE. Les éléments qui ne sont pas utilisés par le concepteur d'IE et qui sont modélisés par l'analyste du domaine de travail sont donc superflus et leur représentation vient inutilement alourdir la tâche de celui-ci.

4.6. Conclusion

Ce chapitre a présenté une description exhaustive de la HAD. Celle-ci s'est soldée par un méta-modèle de la HAD. Il s'agit d'une manière rigoureuse de mettre en relation les éléments manipulés par l'analyste de domaines de travail.

Une recension de la littérature a permis d'identifier deux problèmes associés à la HAD. Le premier problème porte sur la sémantique vague de la HAD. Le deuxième problème porte sur la difficulté d'utiliser la HAD en CIE.

Le méta-modèle présenté dans ce chapitre vient partiellement solutionner le premier problème en spécifiant les éléments de la HAD (vocabulaire) et les règles d'association entre ces derniers (grammaire). Toutefois, le méta-modèle n'indique pas comment créer ou utiliser une HAD; il se limite à définir ce qu'est une HAD.

En ce qui concerne le deuxième problème, le méta-modèle n'apporte aucune solution. Il vient toutefois éclaircir les raisons pour lesquelles la HAD semble insuffisante pour la CIE. En effet, celle-ci doit être accompagnée d'une liste de variables et d'une liste d'équations. Les analyses de correspondance ont démontrée que la HAD uniquement n'a pas un niveau de granularité assez fin pour concevoir une IE. En effet, la HAD se limite aux concepts tandis qu'une IE présente des termes, regroupés autour de concepts, reliés par des opérateurs. De plus, certains éléments de la HAD ne sont pas utilisés en CIE. Leur représentation est donc superflue.

CHAPITRE 5 - DESCRIPTION DU LANGAGE DE MODÉLISATION PROPOSÉ[22]

La littérature en CIE présente la HAD comme étant la seule technique de représentation de domaines de travail. Or, le chapitre 4 présente plusieurs problèmes de compatibilité avec les IE exigeant de recourir à d'autres techniques de représentation, en l'occurrence la liste des variables et la liste des équations. Ce chapitre présente une nouvelle technique de représentation de domaines de travail. Il s'agit d'un langage de modélisation appelé WoDoMoLEID[23] qui permet de représenter d'une manière intégrée tous les éléments d'un domaine de travail nécessaires à la CIE.

Ce chapitre est scindé en trois sections. La première section présente le langage de modélisation proposé. Sa syntaxe abstraite, c'est-à-dire son méta-modèle, et sa syntaxe concrète correspondante y sont décrites. La deuxième section présente le mécanisme d'extension du langage, par exemple, pour ajouter d'autres types d'opérateurs. La troisième section présente une synthèse des éléments abordés dans ce chapitre.

5.1. Description de WoDoMoLEID

Afin d'assurer la meilleure compatibilité possible avec la structure d'une IE, la syntaxe abstraite de WoDoMoLEID s'appuie sur le méta-modèle d'une IE présenté au chapitre 3 avec toutefois quelques différences. Les éléments principaux de ce nouveau langage sont les suivants : terme, opérateur et contrainte. Le méta-modèle de la syntaxe abstraite présenté dans cette section est divisé selon ces derniers.

---

[22] Le contenu de ce chapitre est présenté en partie dans Moïse et Robert (2007a, b). Toutefois, ce chapitre se veut une version revue et améliorée du contenu de ces deux articles.

[23] Se prononce « wodomoleyd ». Acronyme de : ***Work Domain Modeling Language for Ecological Interface Design.***

La syntaxe concrète propose une représentation visuelle, c'est-à-dire une notation symbolique, des éléments définis par la syntaxe abstraite afin de créer et manipuler des modèles. Un symbole particulier est assigné à chaque élément concret du méta-modèle.

5.1.1. Paquetage des termes

Ce paquetage contient les éléments d'un domaine de travail auxquels il est possible d'assigner une valeur. Si la valeur peut être modifiée dans le temps, il s'agit d'une variable. Dans le cas contraire, il s'agit d'une constante. Les termes doivent nécessairement être regroupés par des concepts qui représentent les objets du domaine de travail; un concept est donc propriétaire de termes et/ou de concepts. La figure 5.1 illustre le paquetage des termes. Il s'agit de la même structure du paquetage des concepts et des termes du méta-modèle d'une IE illustré à la figure 3.9.

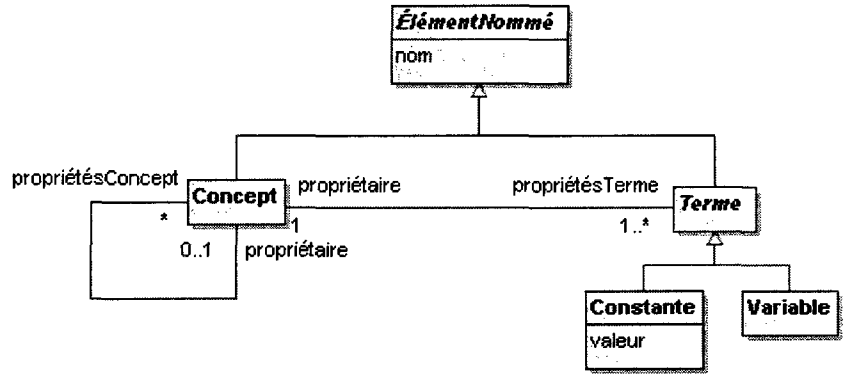


Figure 5.1 – Paquetage des termes.

Les méta-classes `Concept` et `Terme` héritent de la méta-classe abstraite *`ÉlémentNommé`*. Cette dernière définit un élément dont les instances sont identifiées par un nom. Cette généralisation évite de devoir ajouter l'attribut `nom` à différentes méta-classes, évitant ainsi une redondance.

La figure 5.2 illustre la syntaxe concrète de la méta-classe `Concept`. Elle a comme seul attribut nom qui est hérité de la méta-classe abstraite *ÉlémentNommé*. La valeur de ce dernier est une chaîne de caractères écrite en gras et en minuscules, sauf la première lettre qui est en majuscule. Si cette chaîne comporte plusieurs mots, ceux-ci sont concaténés sans espaces et leur première lettre est en majuscule.

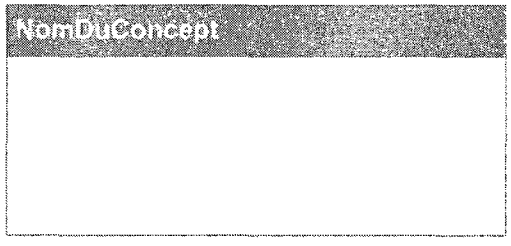


Figure 5.2 – Syntaxe concrète de la méta-classe `Concept`.

La figure 5.3 présente la syntaxe concrète des méta-classes `Variable` et `Constante` qui sont les spécialisations de la méta-classe abstraite *Terme*. L'affichage de la valeur de l'attribut nom qui est hérité de la méta-classe abstraite *ÉlémentNommé* est assujetti aux mêmes règles que la valeur de l'attribut nom d'une instance de la méta-classe `Concept` excepté que la première lettre du premier mot doit être une minuscule. En ce qui concerne une instance de la méta-classe `Constante`, la valeur de l'attribut `valeur` est affichée entre crochets après la valeur de l'attribut nom.

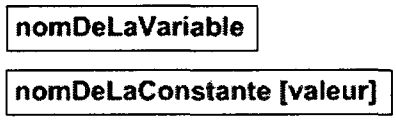


Figure 5.3 – Syntaxe concrète des méta-classes `Variable` et `Constante`.

Les concepts regroupent d'autres concepts et des termes. Cette relation est représentée par l'inclusion à l'intérieur du symbole du concept. Il s'agit de la représentation des

rôles d'associations `propriétaire`, `propriétésConcept` et `propriétésTerme`.

La figure 5.4 présente un exemple d'inclusion. Un concept nommé `Commande` est propriétaire (rôle `propriétaire`) de deux variables (rôle `propriétésTerme`) nommées `date` et `total` et d'un concept (rôle `propriétésConcept`) nommé `LigneCmde`. Ce dernier est propriétaire (rôle `propriétaire`) de deux variables (rôle `propriétésTerme`) nommées `quantité` et `total`. Inversement, ces éléments sont la propriété (rôle `propriétaire`) du concept qui les inclut directement. En effet, la variable `quantité` est la propriété du concept `LigneCmde` et non la propriété du concept `Commande`.

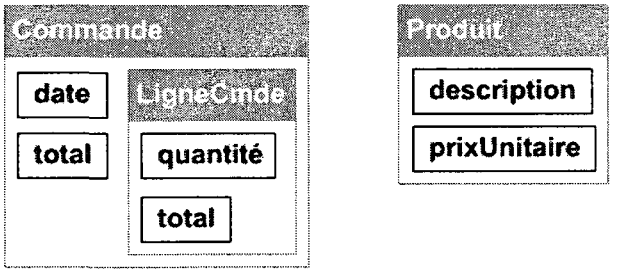


Figure 5.4 – Exemples d'inclusion.

5.1.2. Paquetage des opérateurs

Ce paquetage permet de créer des expressions mathématiques qui servent à définir les contraintes d'un domaine de travail. Il se démarque toutefois du paquetage des expressions du méta-modèle d'une IE illustré à la figure 3.11. Ce dernier permet de rassembler un nombre indéterminé d'opérateurs et de termes dans une seule expression mathématique. Ceci mène à des expressions mathématiques complexes qui peuvent être, d'une part, difficiles à valider et, d'autre part, difficiles à transformer en composants visuels graphiques d'une IE. C'est pour cette raison que le paquetage des opérateurs

impose des règles d'associations limitant le nombre de termes et d'opérateurs d'une expression mathématique. Ces règles sont à la base de l'algorithme de détermination du niveau de chaque contrainte de la hiérarchie fonctionnelle présenté à la section 5.1.3. En somme, plutôt que de définir quelques contraintes avec des expressions mathématiques complexes, WoDoMoLEID privilégie la définition de plusieurs contraintes avec des expressions mathématiques simples.

La figure 5.5 illustre le méta-modèle des opérateurs. La méta-classe abstraite *Opérateur* se spécialise en trois méta-classes abstraites (*Agrégation*, *Arithmétique* et *Logique*) dont chacune représente un type d'opérateur obéissant à des règles particulières d'association avec des termes et/ou des opérateurs. L'attribut `symbole` est hérité par chacune de ces trois méta-classes abstraites.

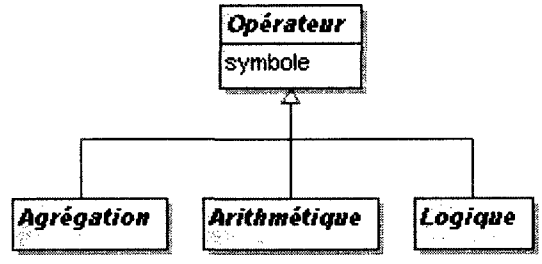


Figure 5.5 – Méta-modèle des opérateurs.

La syntaxe concrète d'un opérateur est illustrée à la figure 5.6. Il s'agit d'un cercle avec au centre le symbole de la relation en gras; le « S » est remplacé par le symbole approprié.

Figure 5.6 – Syntaxe concrète de la méta-classe abstraite `Opérateur`.

Afin de gérer la complexité du paquetage des opérateurs, celui-ci est scindé en trois sous-paquetages distincts; un pour chaque type d'opérateur. Les sections suivantes décrivent chacun de ces sous-paquetages.

5.1.2.1. Agrégation

Un opérateur d'agrégation consiste à faire une opération sur une collection de valeurs d'un même terme. Le résultat de cette opération est représenté par une valeur quantitative. Le langage de modélisation proposé ne fait état que de trois opérateurs d'agrégation : max, min et somme. Le premier consiste à obtenir la valeur maximale de la collection, le deuxième consiste à obtenir la valeur minimale de la collection et le troisième consiste à additionner toutes les valeurs de la collection. La figure 5.7 illustre des exemples d'application d'opérateurs d'agrégation.

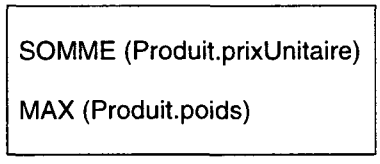


Figure 5.7 – Exemples d'application d'opérateurs d'agrégation.

Le paquetage des opérateurs d'agrégation est illustré à la figure 5.8. Une instance de la méta-classe abstraite `Agrégation` est obligatoirement associée à une instance de la méta-classe abstraite `Terme` qui représente l'opérande associé à l'opérateur.

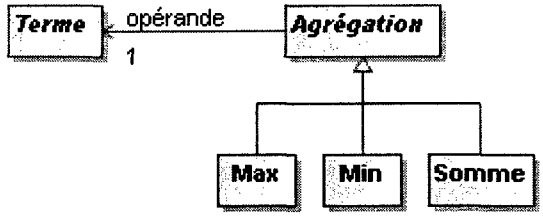


Figure 5.8 – Méta-modèle des opérateurs d'agrégation.

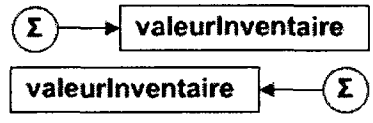
Bien que d'autres opérateurs de ce type existent, par exemple compte et moyenne, ils ont été omis pour des raisons de simplicité. Il est toutefois possible de les ajouter au méta-modèle par le mécanisme décrit à la section 5.2.

Le tableau 5.1 présente les symboles de chaque type d'opérateur d'agrégation du méta-modèle de la figure 5.8.

Tableau 5.1 – Syntaxe concrète de chaque opérateur d'agrégation.

Méta-classe concrète	Symbole
Max	▲
Min	▼
Somme	Σ

La figure 5.9 illustre un exemple de la syntaxe concrète de la méta-classe `Somme`. La flèche avec une pointe fermée et pleine représente le rôle d'association `opérande`.



`SOMME(valeurInventaire)`

Figure 5.9 – Exemple de syntaxe concrète de l'opérateur d'agrégation.

Il est important de noter que selon la syntaxe abstraite, un opérateur « connaît » l'opérande auquel il est associé. Toutefois, ce dernier ne « connaît » pas l'opérateur auquel il est associé. En somme, la manière de lire les associations est de fixer en premier lieu un opérateur et ensuite d'identifier son opérande en suivant la flèche.

5.1.2.2. *Arithmétique*

Un opérateur arithmétique consiste à réaliser une opération sur deux variables quantitatives appelées opérandes. Le résultat de cette opération est représenté par une valeur quantitative. Pour les opérateurs arithmétiques de soustraction et de division, puisqu'ils sont non commutatifs contrairement à l'addition et à la multiplication, l'ordre des opérandes est important. Bien qu'une expression arithmétique puisse comporter plusieurs opérateurs arithmétiques, le langage de modélisation proposé limite à un seul opérateur afin d'éviter les expressions complexes. La figure 5.10 illustre des exemples d'application d'opérateurs arithmétiques.

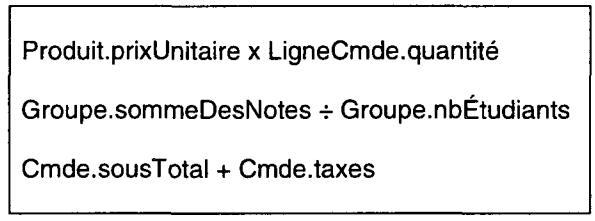


Figure 5.10 – Exemples d'application d'opérateurs arithmétiques.

Le méta-modèle des opérateurs arithmétiques est illustré à la figure 5.11. Une instance de la méta-classe abstraite `Arithmétique` est obligatoirement associée à deux instances distinctes de la méta-classe abstraite `Terme`, l'une représentant l'opérande de gauche et l'autre représentant l'opérande de droite. Il ne peut toutefois s'agir de la même instance de la méta-classe abstraite `Terme`.

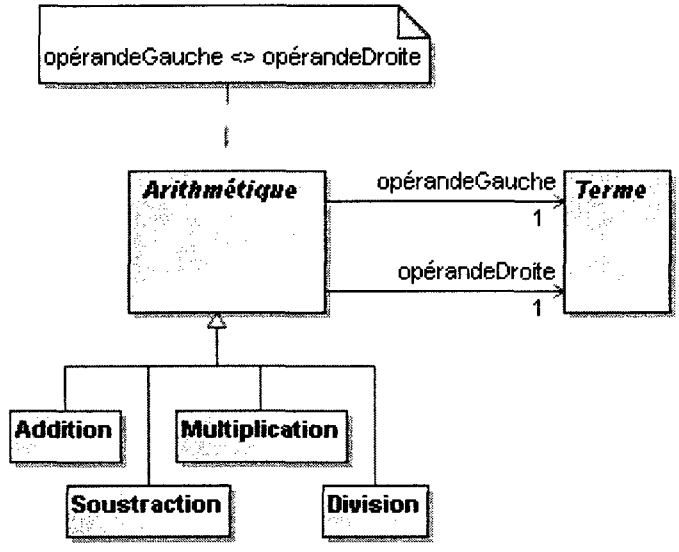


Figure 5.11 – Méta-modèle des opérateurs arithmétiques.

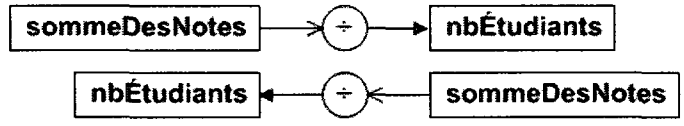
Bien que d'autres opérateurs de ce type existent, par exemple l'exposant, ils ont été omis pour des raisons de simplicité. Il est toutefois possible de les ajouter au méta-modèle par le mécanisme décrit à la section 5.2.

Le tableau 5.2 présente les symboles de chaque type d'opérateur arithmétiques du méta-modèle de la figure 5.11.

Tableau 5.2 – Syntaxe concrète de chaque opérateur arithmétique.

Méta-classe concrète	Symbole
Addition	+
Division	÷
Multiplication	x
Soustraction	-

La figure 5.12 illustre un exemple de la syntaxe concrète de la méta-classe `Division`. La flèche avec la pointe ouverte représente l'association `opérandeGauche` et la flèche avec la pointe fermée et pleine représente l'association `opérandeDroite`. Ainsi, quelle que soit la position des flèches, la lecture se fait toujours de la même manière, c'est-à-dire dans l'ordre des flèches. Par conséquent, les deux représentations de la division sont équivalentes.



`sommeDesNotes ÷ nbÉtudiants`

Figure 5.12 – Exemple de syntaxe concrète de l'opérateur arithmétique.

Dans la même veine que pour l'opérateur d'agrégation, un opérateur « connaît » les opérandes auxquels il est associé. Toutefois, ces dernières ne « connaissent » pas l'opérateur auquel ils sont associés. Encore une fois, la manière de lire les associations est de fixer en premier lieu un opérateur et ensuite d'identifier ses opérandes en suivant les flèches.

5.1.2.3. Logique

Un opérateur logique consiste à réaliser une opération sur deux opérandes. Le résultat de cette opération est représenté par une valeur booléenne, c'est-à-dire qui ne peut prendre qu'une seule de deux valeurs : vrai ou faux. Le langage de modélisation proposé impose une instance de la méta-classe abstraite *`Terme`* comme opérande de gauche. En ce qui concerne l'opérande de droite, celui-ci peut être une instance de la méta-classe abstraite *`Terme`*, une instance de la méta-classe abstraite *`Agrégation`* ou une instance de la méta-classe abstraite *`Arithmétique`*. La figure 5.13 illustre des exemples d'application d'opérateurs logiques.

Produit.valeurInventaire = Produit.coûtUnitaire x Produit.qteEnStock

Inventaire.valeur = Σ (Produit.valeurInventaire)

Inventaire.valeurMax > Inventaire.valeur

Figure 5.13 – Exemples d'application d'opérateurs logiques.

Le paquetage des opérateurs logiques est illustré à la figure 5.14. Une instance de la méta-classe abstraite `Logique` est obligatoirement associée à une seule instance de la méta-classe `Terme` en tant qu'opérande de gauche. En ce qui concerne l'opérande de droite, il est associé à une méta-classe abstraite `CôtéDroit` qui peut représenter un opérateur d'agrégation, un opérateur arithmétique ou un terme. L'instance qui représente l'opérande de gauche doit être différente de l'instance qui représente l'opérande de droite.

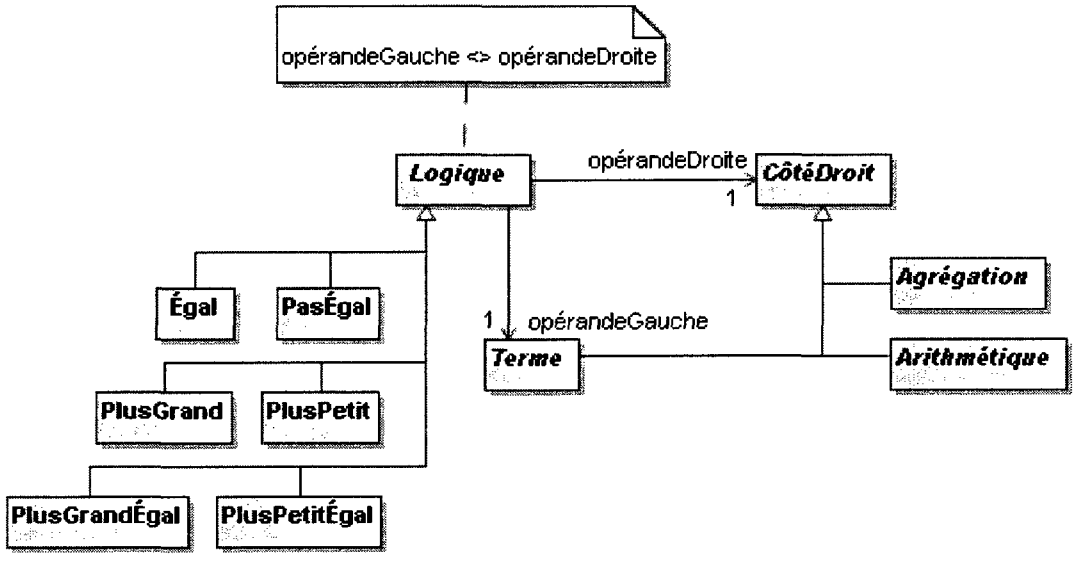


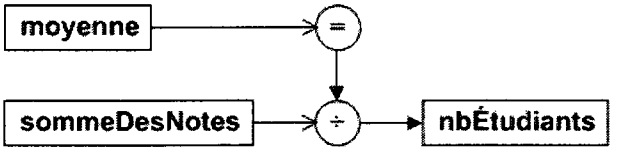
Figure 5.14 – Méta-modèle des opérateurs logiques.

Le tableau 5.3 présente les symboles de chaque type d'opérateur logique du méta-modèle de la figure 5.14.

Tableau 5.3 – Syntaxe concrète de chaque opérateur logique.

Méta-classe concrète	Symbole
Égal	=
Pas égal	≠
Plus grand	>
Plus petit	<
Plus grand ou égal	≥
Plus petit ou égal	≤

La figure 5.15 illustre un exemple de la syntaxe concrète de la méta-classe `Égal`. Tout comme pour l'opérateur arithmétique, la flèche avec la pointe ouverte représente l'association `opérandeGauche` et la flèche avec la pointe fermée et pleine représente l'association `opérandeDroite`. Ainsi, quelle que soit la position des flèches, la lecture se fait toujours de la même manière, c'est-à-dire dans l'ordre des flèches.




```
moyenne = sommeDesNotes ÷ nbÉtudiants
```

Figure 5.15 – Exemple de syntaxe concrète de l'opérateur logique.

La représentation de l'association entre un opérateur et ses opérandes est plus complexe pour les opérateurs logiques que pour les opérateurs arithmétiques. Dans le cas de la figure 5.15, l'opérande de gauche est une instance de la méta-classe `Variable`, une

méta-classe qui hérite de la méta-classe abstraite *Terme*, et l'opérande de droite est une instance de la méta-classe abstraite *CôtéDroit*, plus précisément une instance de la méta-classe `Division`, une méta-classe qui hérite de la méta-classe abstraite *Arithmétique*.

Tout comme pour les autres types d'opérateurs, celui-ci « connaît » les opérandes auxquels il est associé, mais les opérandes ne « connaissent » pas l'opérateur auquel ils sont associés. Il faut donc, encore une fois, s'en remettre au sens des flèches. Par exemple, à la figure 5.15, l'opérateur de division est associé à deux variables qui sont situées à gauche et à droite et qui représentent respectivement l'opérande de gauche et l'opérande de droite. La flèche du haut n'est pas à considérer dans ce cas car elle indique l'opérande de droite de l'opérateur d'égalité. En tant qu'opérande, l'opérateur de division n'a pas à « connaître » l'opérateur auquel il est associé, en l'occurrence l'opérateur d'égalité.

5.1.3. Paquetage des contraintes

Ce paquetage permet de définir un ensemble de contraintes dont les expressions mathématiques associées vont constituer la hiérarchie fonctionnelle. La figure 5.16 illustre le méta-modèle des contraintes.

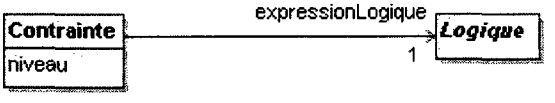


Figure 5.16 – Méta-modèle des contraintes.

La méta-classe `Contrainte` possède un attribut `niveau` qui identifie le niveau de la contrainte dans la hiérarchie fonctionnelle. Moins cette valeur est élevée, plus elle s'approche de la description des buts du système, c'est-à-dire sa raison d'être; une

contrainte de niveau 1 représente un but du système. La hiérarchie fonctionnelle est obtenue par la structure des relations entre les termes utilisés à la fois comme opérande de gauche pour une contrainte et comme opérande de droite pour une autre[24]. La figure 5.17 présente l'algorithme permettant d'identifier le niveau de chaque contrainte.

Cet algorithme est réalisé en trois opérations et présuppose une liste d'objets de type *Contrainte* nommée `listeContraintes`. La première opération, appelée `assignerNiveaux()`, consiste à passer en boucle chaque élément de `listeContraintes` et assigner leur niveau en appelant l'opération `trouverNiveau()`.

L'opération `trouverNiveau()` calcule la « profondeur » d'une contrainte dans la hiérarchie fonctionnelle. Elle consiste premièrement, pour une contrainte, à sélectionner l'opérande de gauche de l'opérateur logique associé. S'il n'y en a pas, la boucle se termine et la valeur de l'attribut `niveau` est retournée. S'il existe un opérande de gauche pour cette contrainte, il faut trouver la contrainte dont l'opérateur logique contient cet opérande, mais dans le côté droit. Ceci est réalisé par l'opération `trouverContrainte()`.

L'opération `trouverContrainte()` permet, à partir d'un terme, de trouver la contrainte où celui-ci fait partie du côté droit de l'opérateur logique associé à cette contrainte. Dépendamment du type de l'objet qui représente le côté droit de l'opérateur logique (*Terme*, *Agrégation* ou *Arithmétique*), la vérification se fait d'une manière différente.

---

[24] Le méta-modèle fait en sorte que, dans une hiérarchie fonctionnelle, un même terme me peut être utilisé qu'une seule fois comme opérande de gauche d'une expression logique et ne peut être contenu qu'une seule fois dans le côté droit d'une expression logique.

```
Liste listeContraintes

assignerNiveaux() {
        POUR CHAQUE contrainte DE listeContraintes
                contrainte.niveau ← trouverNiveau(contrainte)
}

trouverNiveau(Contrainte contrainte) {
        niveau ← 0
        TANT QUE contrainte ≠ null
                niveau ← niveau + 1
                Terme opérandeGauche ← contrainte.opérateurLogique.opérandeGauche
                SI opérandeGauche ≠ null
                        contrainte ← trouverContrainte(opérandeGauche)
                SINON
                        contrainte ← null

        retourner niveau
}

trouverContrainte(Terme terme) {
        Contrainte contrainteCible ← null
        POUR CHAQUE contrainte DE listeContraintes
                CôtéDroit côtéDroit ← contrainte.opérateurLogique.côtéDroit
                SI côtéDroit ≠ null
                        SI côtéDroit EST Terme
                                termeDroite ← côtéDroit
                                SI termeDroite = terme
                                        contrainteCible ← contrainte
                        SINON SI côtéDroit EST Agrégation
                                opération ← côtéDroit
                                SI(opération.opérande = terme)
                                        contrainteCible ← contrainte
                        SINON SI côtéDroit EST Arithmétique
                                opération ← côtéDroit
                                SI(opération.opérandeGauche = terme OU opération.opérandeDroite = terme)
                                        contrainteCible ← contrainte

        retourner contrainteCible
}
```

Figure 5.17 – Algorithme permettant d'identifier le niveau de chaque contrainte.

La figure 5.18 présente un ensemble de contraintes et la hiérarchie fonctionnelle sous-jacente. Il s'agit de la hiérarchie fonctionnelle présentée à la figure 3.14 à laquelle l'identification des niveaux de chaque contrainte a été ajoutée.

Niveau	Contrainte
1	objectif < profits
2	profits = revenus - dépenses
3	revenus = prixUnitaire * qtéVendue
3	dépenses = fraisFixes + fraisVariables
4	fraisVariables = fraisUnitaire * nbUnitésProduites

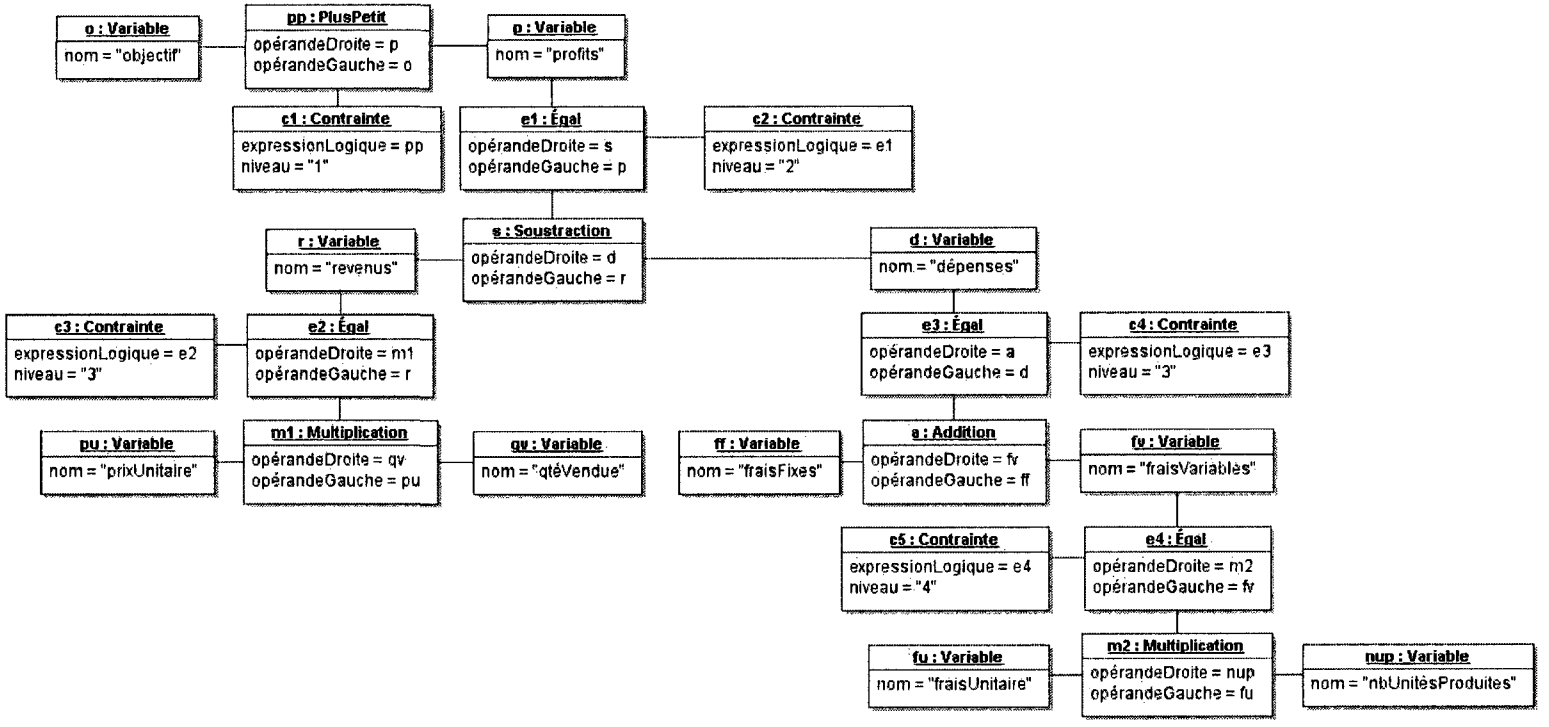


Figure 5.18 – Hiérarchie fonctionnelle sous-jacente à un ensemble de contraintes avec leur niveau.

La figure 5.19 illustre un exemple de la syntaxe concrète de la méta-classe `Contrainte`. Une contrainte est représentée textuellement avec son niveau et l'opération logique associée.

```
2    profits = revenus - dépenses
```

Figure 5.19 – Exemple de syntaxe concrète d'une contrainte.

5.2. Extension de la syntaxe abstraite

La syntaxe abstraite du langage de modélisation proposé est flexible en ce sens qu'il est possible d'ajouter des méta-classes. Par exemple, si nécessaire, il serait possible d'ajouter les méta-classes `Moyenne` comme spécialisations de la méta-classe abstraite *`Agrégation`* et `Exposant` comme spécialisations de la méta-classe abstraite *`Arithmétique`*. La première opération consiste à obtenir la moyenne d'une collection de valeurs. La deuxième opération consiste à obtenir le résultat de l'opérande de gauche multiplié par lui-même un certain nombre de fois tel que déterminé par l'opérande de droite. La figure 5.20 illustre cette situation.

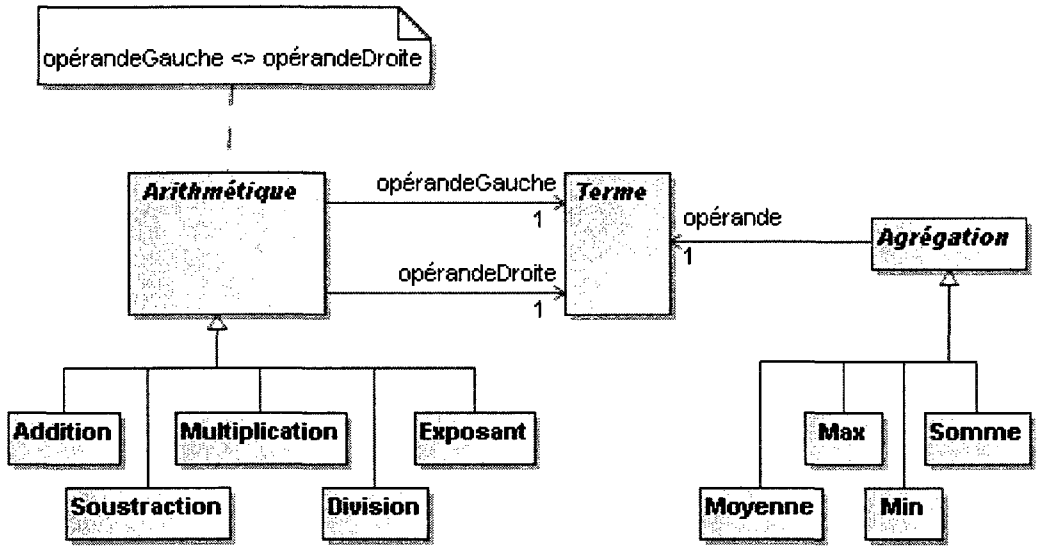


Figure 5.20 – Extension du méta-modèle.

Une fois positionnées dans le méta-modèle, ces nouvelles méta-classes doivent être associées à une syntaxe concrète. Le tableau 5.4 présente leur symbole affiché dans le cercle (syntaxe concrète de l'opérateur).

Tableau 5.4 – Syntaxe concrète des nouveaux opérateurs.

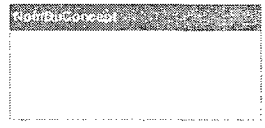
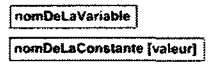
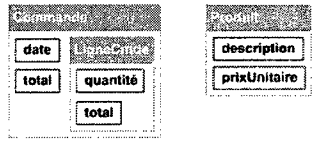
Méta-classe concrète	Symbole
Exposant	^
Moyenne	$\bar{x}$

5.3. Synthèse

Ce chapitre présente WoDoMoLEID, un langage de modélisation de domaines de travail pour la conception d'IE. Plus particulièrement, la syntaxe abstraite et la syntaxe concrète sont décrites.

Le méta-modèle de WoDoMoLEID se divise en trois paquetages : termes, opérateurs et contraintes. Le premier paquetage porte sur l'identification des objets (concepts) et des informations (termes) du domaine de travail. Le deuxième paquetage porte sur l'identification des relations entre les informations. Le troisième paquetage permet de rassembler les termes et les opérateurs en contraintes. Ces dernières constituent la pierre angulaire de la CIE. Le tableau 5.5 présente une synthèse des principaux éléments de WoDoMoLEID.

Tableau 5.5 – Synthèse des éléments principaux de WoDoMoLEID.

Élément	Symbole
Concept	
Terme	
propriétaire propriétésConcept propriétésTerme	
Opérateur	
opérande opérandeDroite	
opérandeGauche	

Afin de permettre une correspondance directe entre les éléments du domaine de travail et ceux d'une IE, le méta-modèle de WoDoMoLEID s'appuie sur celui des IE. Avec la HAD et la liste des équations, qui représentent les contraintes du domaine de travail, les contraintes sont écrites librement sous forme d'équations mathématiques par l'analyste. Toutefois, les composants visuels réutilisables proposés par Burns et Hajdukiewicz

(2004), qui représentent graphiquement les contraintes du domaine de travail, s'appuient sur des règles précises de relations entre des termes et des opérateurs. En d'autres mots, les contraintes doivent s'appuyer sur une structure spécifique pour correspondre à des composants visuels réutilisables. Le chapitre suivant présente l'évaluation de WoDoMoLEID tant en ce qui porte sur sa capacité à représenter un domaine de travail que de permettre une correspondance directe avec des composants visuels réutilisables.

CHAPITRE 6 - ÉVALUATION DU LANGAGE DE MODÉLISATION PROPOSÉ

Le chapitre 5 présente un langage de modélisation de contraintes de domaines de travail, appelé WoDoMoLEID, qui s'appuie sur le méta-modèle des IE. La description de ce langage de modélisation permet d'atteindre partiellement le deuxième objectif de cette thèse qui est de créer et évaluer un langage de modélisation de domaines de travail quantitatifs permettant une correspondance directe avec les composants visuels d'une interface écologique. Ce chapitre permet de faire le reste du chemin en présentant une évaluation du langage de modélisation.

L'évaluation est scindée en deux parties. La première partie porte sur la capacité du langage à soutenir la conception d'IE. La deuxième partie porte sur la capacité du langage à permettre une correspondance directe entre les contraintes modélisées à l'aide de ce dernier et les composants visuels qui composent une IE.

6.1. Soutenir la conception d'interfaces écologiques

Le chapitre 3 présente une description détaillée des IE, un type particulier d'IU. Leur conception s'appuie sur trois principes dont chacun est associé au soutien d'un des trois niveaux de contrôle cognitif de la taxonomie SRK (Rasmussen 1983) :

- **SBB.** L'individu doit pouvoir manipuler les objets du domaine de travail directement sur l'IU.
- **RBB.** Une correspondance un-à-un doit exister entre les contraintes du domaine de travail et les informations présentées par l'IU.
- **KBB.** L'IU doit présenter la hiérarchie fonctionnelle du domaine de travail.

Un langage de modélisation de domaines de travail doit donc permettre d'identifier tous les éléments associés à chacun des trois niveaux de contrôle cognitif. Particulièrement, celui-ci doit permettre de définir les éléments suivants appartenant au domaine de travail : (a) les objets, (b) les contraintes et (c) la hiérarchie fonctionnelle.

Afin d'évaluer dans ce sens le langage proposé dans cette thèse, il est nécessaire de le mettre à l'épreuve. Pour ce faire, un prototype de logiciel de modélisation à été développé. Celui-ci implémente le méta-modèle de WoDoMoLEID présenté au chapitre 5. Le logiciel permet de réaliser des modèles qui respectent les règles définies par le méta-modèle. La section 6.1.1 présente une description du prototype de logiciel de modélisation.

La section 6.1.2 présente une application de WoDoMoLEID à la gestion de portefeuille d'actifs financiers réalisée avec le prototype de logiciel de modélisation présenté à la section 6.1.1. Ce cas d'application permet d'évaluer la capacité de ce langage à modéliser les éléments d'un domaine de travail pour des fins de conception d'IE. Une discussion de cette évaluation est présentée à la section 6.1.3.

6.1.1. Prototype de logiciel de modélisation

Un modèle n'a pas besoin d'être réalisé par un logiciel; il peut être réalisé simplement en utilisant un crayon et du papier. Toutefois, un logiciel de modélisation fournit un environnement permettant, entre autres, de renforcer les règles d'un langage (Kelly et Tolvanen 2008). Par exemple, un logiciel peut indiquer si le modèle, ou un sous-ensemble de celui-ci, est incomplet et peut empêcher les associations non permises entre certains éléments. En somme, un logiciel de modélisation implémente la syntaxe abstraite et la syntaxe concrète d'un langage de modélisation.

Une autre raison pour le développement d'un prototype de logiciel de modélisation est la validation de l'algorithme présenté à la section 5.1.3 (figure 5.9) qui permet d'identifier le niveau de chaque contrainte. Cet algorithme est implémenté dans le logiciel et devrait permettre d'assigner automatiquement chaque contrainte à un niveau particulier.

La figure 6.1 présente l'IU du logiciel de modélisation. Celui-ci comporte quatre zones. La première zone, en haut à l'extrême gauche, est la barre d'outils. Celle-ci permet de sélectionner les fonctionnalités de création, modification et suppression des différents symboles du langage de modélisation. La deuxième zone, en haut au centre, est le canevas de modélisation. C'est sur le canevas que l'utilisateur peut créer un modèle à partir des éléments de la barre d'outils. La troisième zone, en bas à gauche, est celle des avertissements. Cette zone indique qu'une ou plusieurs règles de modélisation sont enfreintes, par exemple un terme qui n'est pas associé à un concept ou une contrainte incomplète. La quatrième zone, en bas à droite, est la hiérarchie des contraintes. Elle affiche le résultat de l'algorithme permettant d'identifier le niveau de chaque contrainte, en d'autres mots, la hiérarchie fonctionnelle.

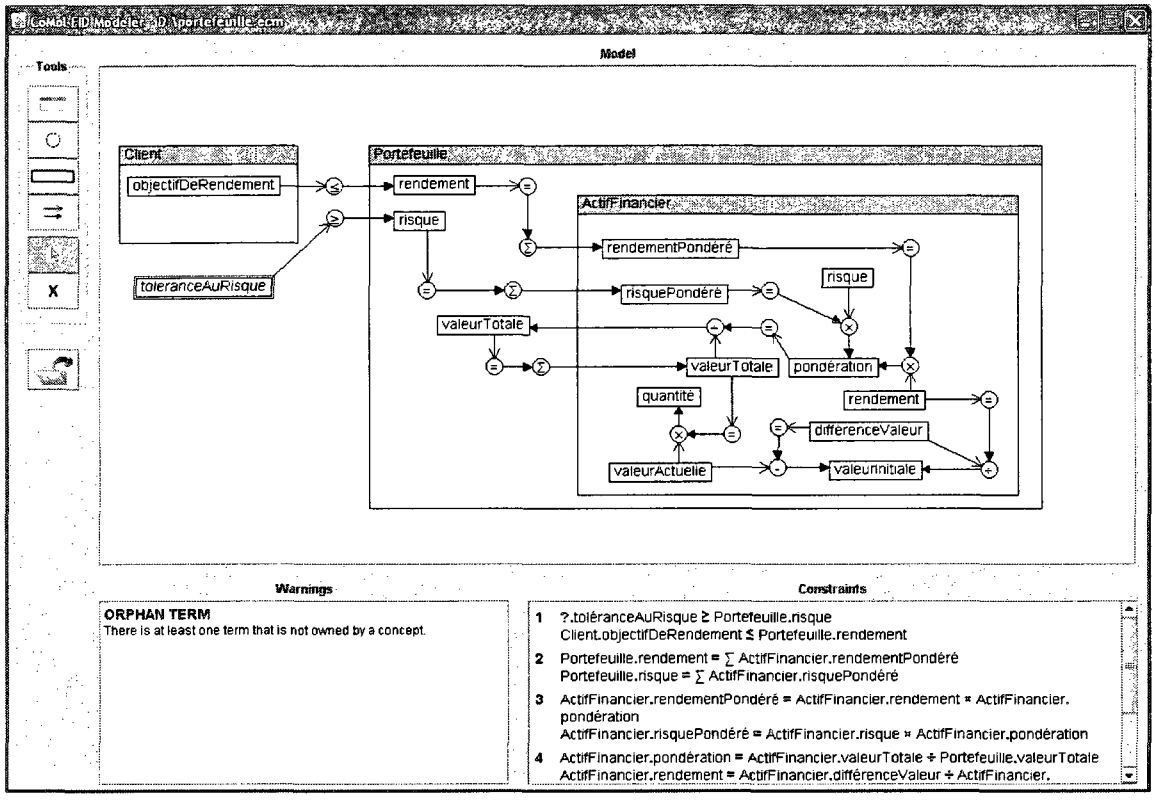


Figure 6.1 – Interface utilisateur du prototype de logiciel de modélisation.

6.1.2. Application à la gestion de portefeuille d'actifs financiers

Un portefeuille d'actifs financiers peut être considéré comme un système dont les objectifs sont (a) de maximiser le rendement et (b) de préserver le niveau de risque sous un certain seuil. Ces deux objectifs sont conflictuels en ce sens qu'ils imposent des restrictions l'un sur l'autre. D'une part, la maximisation du rendement est limitée par le seuil de risque; pour un niveau de risque donné, il est impossible d'aller au-delà d'un certain niveau de rendement. D'autre part, la réduction du risque ne permet pas d'atteindre le rendement maximum; à cause de la relation risque-rendement, réduire le seuil de risque implique une réduction du rendement atteignable. En somme, il s'agit d'optimiser le rendement en fonction du risque.

Ce système est régi par un ensemble de contraintes pouvant être représentées sous la forme d'équations mathématiques. La valeur du rendement et du risque du portefeuille est influée par la valeur actuelle, la valeur à l'achat, le risque et la quantité de chaque actif financier. Le gestionnaire de portefeuille peut contrôler uniquement la quantité de chaque actif financier.

Ce cas est entièrement quantitatif, il contient deux objectifs conflictuels, un nombre de concepts (3), de termes (15) et de contraintes (11) assez élevé pour ne pas être trop simpliste, mais assez bas pour ne pas être trop complexe. Par conséquent, il est approprié pour la présente évaluation.

La figure 6.2 illustre le modèle du domaine de travail, réalisé à l'aide de WoDoMoLEID, pour le cas de gestion de portefeuille d'actifs financiers.

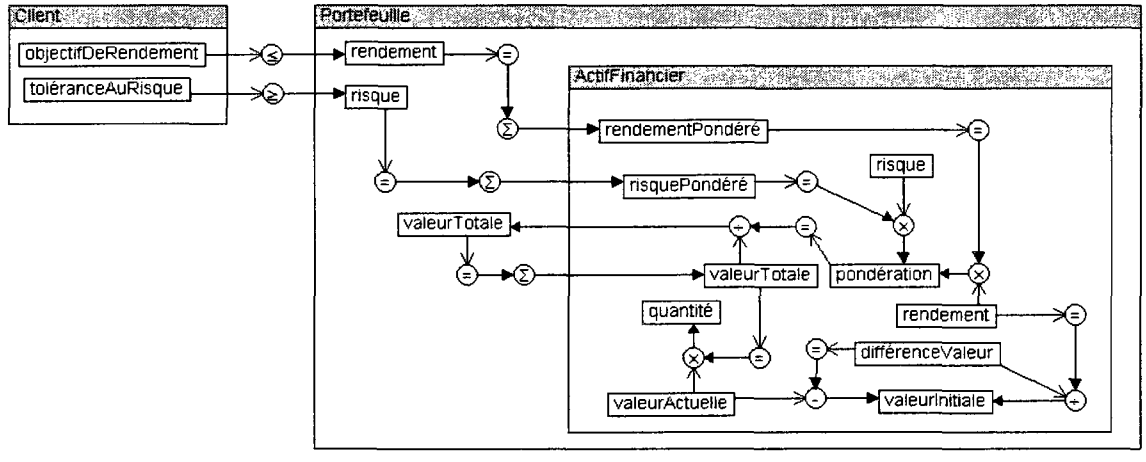


Figure 6.2 – Modèle du domaine de travail.

La figure 6.3 présente l'ensemble des contraintes du domaine de travail. Chacune d'entre elle est associée à un niveau d'abstraction particulier. Cet ensemble est dérivé du modèle du domaine de travail illustré à la figure 6.2.

1 Client.toléranceAuRisque ≥ Portefeuille.risque
Client.objectifDeRendement ≤ Portefeuille.rendement

2 Portefeuille.rendement = ∑ ActifFinancier.rendementPondéré
Portefeuille.risque = ∑ ActifFinancier.risquePondéré

3 ActifFinancier.rendementPondéré = ActifFinancier.rendement × ActifFinancier.pondération
ActifFinancier.risquePondéré = ActifFinancier.risque × ActifFinancier.pondération

4 ActifFinancier.pondération = ActifFinancier.valeurTotale ÷ Portefeuille.valeurTotale
ActifFinancier.rendement = ActifFinancier.différenceValeur ÷ ActifFinancier.valeurInitiale

5 ActifFinancier.différenceValeur = ActifFinancier.valeurActuelle - ActifFinancier.valeurInitiale
Portefeuille.valeurTotale = ∑ ActifFinancier.valeurTotale
ActifFinancier.valeurTotale = ActifFinancier.valeurActuelle × ActifFinancier.quantité

Figure 6.3 – Contraintes par niveau d'abstraction.

6.1.2.1. Soutenir le SBB

À la lumière des figures 6.2 et 6.3, le langage de modélisation permet d'identifier les objets du domaine de travail qui doivent être présentés par l'IU. Ces objets sont représentés par les concepts suivants : client, portefeuille et actif financier. Tous les termes sont associés à un de ces concepts; ils servent à définir ceux-ci, par exemple, le client est défini par son objectif de rendement et sa tolérance au risque.

6.1.2.2. Soutenir le RBB

La particularité d'une IE par rapport à une IU traditionnelle est le fait qu'elle met l'accent non pas sur la présentation d'informations mais sur les relations entre celles-ci. Dans le cas d'un domaine de travail de nature quantitative, ces relations sont définies par des opérateurs mathématiques reliés à des termes. Lorsque l'opérateur est de type logique, l'expression mathématique correspond à une contrainte.

Les figures 6.2 et 6.3 démontrent que le langage de modélisation permet d'identifier les contraintes d'un domaine de travail. Chaque contrainte est définie par une expression logique pouvant prendre la valeur « vrai » ou « faux ». Tant que la valeur de chaque

contrainte est « vrai », le système est dans un état normal. Si la valeur d'une contrainte est « faux », celle-ci est brisée et le système est dans un état anormal.

6.1.2.3. Soutenir le KBB

La hiérarchie fonctionnelle peut être définie comme étant l'ensemble inter-relié de contraintes d'un domaine de travail. Il s'agit de l'ensemble de tous les termes qui sert à représenter le domaine de travail à travers différents niveaux d'abstraction.

La hiérarchie fonctionnelle peut être dérivée à l'aide de l'algorithme présenté à la section 5.1.3. Cet algorithme permet d'assigner un niveau d'abstraction spécifique à une contrainte en fonction de sa « profondeur » dans la hiérarchie fonctionnelle. La figure 6.3 présente la hiérarchie fonctionnelle qui s'appuie sur le modèle illustré à la figure 6.2. La valeur de chaque terme du côté droit de chaque contrainte est expliquée par une contrainte de niveau inférieur.

Les informations présentées à la figure 6.3 peuvent être réorganisées pour faire ressortir davantage les relations entre les termes associés à un des cinq niveaux d'abstraction. Le résultat est illustré à la figure 6.4 qui présente la hiérarchie fonctionnelle des termes. Chacune des relations de cette structure est définie par un opérateur mathématique lorsque modélisées à l'aide de WoDoMoLEID. Il est important de garder à l'esprit que ce sont les contraintes qui sont associées aux niveaux d'abstraction et non les termes. C'est pour cette raison que les termes `ActifFinancier.valeurTotale` et `ActifFinancier.valeurInitiale` chevauchent deux niveaux d'abstraction; ces termes sont utilisés à la fois par des contraintes de niveau 4 et 5. Quatre termes ne sont pas reliés à des termes de niveaux inférieurs : `ActifFinancier.quantité`, `ActifFinancier.valeurActuelle`, `ActifFinancier.valeurInitiale` et `ActifFinancier.risque`. La raison est que la valeur de ces termes est obtenue

soit par l'utilisateur de l'IE (`ActifFinancier.quantité`), soit par une source externe (`ActifFinancier.valeurActuelle`, `ActifFinancier.valeurInitiale` et `ActifFinancier.risque`), par exemple un fournisseur de données financières dans ce cas.

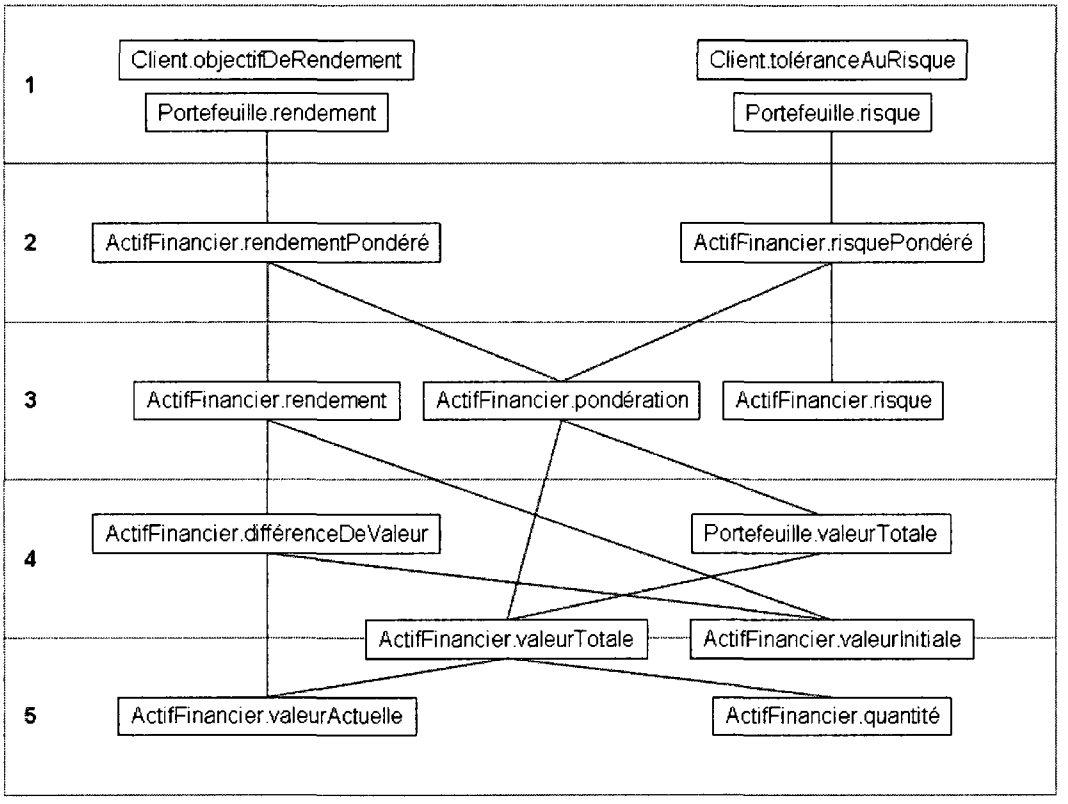


Figure 6.4 – Hiérarchie fonctionnelle des termes.

La hiérarchie fonctionnelle illustrée à la figure 6.4 s'apparente à la structure de la HAD. Est-il possible de présenter cette hiérarchie fonctionnelle de termes dans une HAD? Si oui, c'est signe que WoDoMoLEID est un équivalent à la HAD pour définir la hiérarchie fonctionnelle d'un domaine de travail. Par conséquent, WoDoMoLEID serait conforme à la philosophie écologique d'analyse de domaines de travail.

La figure 6.5 illustre la transposition de la hiérarchie fonctionnelle de la figure 6.4 en une HAD. Chaque terme peut être associé à une case, c'est-à-dire à un niveau d'abstraction et à un niveau de décomposition, sauf pour deux termes qui sont associés à deux niveaux d'abstraction à la fois. Malgré cela, ces termes, qui ont été définis en utilisant WoDoMoLEID, constituent une hiérarchie fonctionnelle compatible avec la HAD. WoDoMoLEID permet donc de représenter la même hiérarchie fonctionnelle d'un domaine de travail que si elle avait été obtenue en utilisant une HAD.

	Client	Portefeuille	Actif financier
1	objectifDeRendement toléranceAuRisque	rendement risque	
2			rendementPondéré risquePondéré
3			rendement pondération risque
4		valeurTotale	différenceDeValeur
4-5			valeurTotale valeurInitiale
5			valeurActuelle quantité

Figure 6.5 – Hiérarchie d'abstraction et de décomposition.

6.1.3. Discussion

La section 4.3 présente deux problèmes associés à la HAD. Le premier problème porte sur sa sémantique imprécise et le deuxième problème porte sur sa portée limitée pour la CIE. Cette section discute les résultats de la présente évaluation en lien avec ces deux problèmes.

La sémantique imprécise de la HAD affecte autant la dimension d'abstraction que la dimension de décomposition. En ce qui concerne la dimension d'abstraction, le problème est relié à la définition des niveaux d'abstraction sous-jacents à la hiérarchie fonctionnelle. WoDoMoLEID solutionne ce problème en permettant, à l'aide d'un algorithme à cet effet, de dériver les niveaux d'abstraction. Plus particulièrement, au fur et à mesure que les contraintes sont définies à l'aide du langage, l'algorithme permet de les assigner à un niveau particulier. Ainsi, la hiérarchie fonctionnelle est obtenue automatiquement sans avoir à se soucier de définir un nombre de niveaux d'abstraction et de les nommer comme c'est le cas pour la HAD.

En ce qui concerne la dimension de décomposition, la HAD ne distingue pas la collaboration de la décomposition. WoDoMoLEID présente une syntaxe concrète beaucoup plus précise à ce sujet. En effet, il est possible de voir à la figure 6.2 la collaboration entre le concept `Client` et le concept `Portefeuille` et la décomposition[25] entre le concept `Portefeuille` et le concept `ActifFinancier`.

Tel que démontré à la section 4.5, la HAD ne permet pas de modéliser un domaine de travail de manière à identifier les informations pertinentes à la CIE, c'est-à-dire les objets, les contraintes et la hiérarchie fonctionnelle d'un domaine de travail. Pour ce faire, il faut l'accompagner d'une liste des variables et d'une liste des équations, cette dernière étant le document principal pour la CIE. En somme, la HAD ne présente qu'une vue de haut niveau de la hiérarchie d'abstraction. La présente évaluation a démontré que WoDoMoLEID permet d'identifier toutes les informations nécessaires à la CIE, et ce, d'une manière intégrée.

[25] La décomposition fait référence à l'inclusion. Il est possible de dire qu'un portefeuille se *décompose* en actifs financiers ou qu'un portefeuille *inclut* des actifs financiers.

6.2. Correspondance entre les contraintes et les composants visuels

Tel que présenté au chapitre 4, la HAD est une technique de représentation de domaines de travail qui consiste à représenter des concepts inter-reliés à différents niveaux d'abstraction. Cette structure de concepts constitue une vue de haut niveau de la hiérarchie fonctionnelle. Toutefois, la HAD ne permet pas de décrire les contraintes du domaine de travail qui sont essentiels à la CIE. Pour palier ce problème, un analyste de domaines de travail doit, en s'appuyant sur la hiérarchie fonctionnelle obtenue en utilisant la HAD, créer une liste de variables et une liste d'équations.

Burns et Hajdukiewicz (2004) ont proposé un thésaurus visuel, c'est-à-dire un catalogue de composants visuels réutilisables pour la CIE. Chaque composant visuel représente un type particulier de contrainte. Le problème est que chaque composant visuel s'appuie sur une structure spécifique de contraintes. Ainsi, si les contraintes définies par l'analyste ne respectent pas cette structure, la correspondance avec les composants visuels ne peut pas être directe; le concepteur doit redéfinir les contraintes afin qu'elles correspondent à la structure des composants visuels. Par conséquent, il est difficile, voire impossible, d'automatiser le processus de sélection des composants visuels si les contraintes sont écrites librement.

Pour résoudre ce problème, WoDoMoLEID impose une structure dans la manière de définir les contraintes d'un domaine de travail qui est compatible avec la structure de composants visuels prédéfinis. Cette section consiste à évaluer la capacité de WoDoMoLEID à cet effet. Pour ce faire, la section 6.2.1 présente un catalogue de composants visuels. Chaque composant visuel est associé à une manière de définir un type de contrainte avec WoDoMoLEID. La section 6.2.2 présente une application à la gestion de portefeuille d'actifs financiers. Une maquette d'IE, intégrant les composants

visuels sélectionnés, est présentée. La section 6.2.3 présente une discussion de cette évaluation.

6.2.1. Catalogue de composants visuels

Cette section présente un catalogue de composants visuels réutilisables pour la CIE. Un composant visuel est un agencement de formes géométriques qui représente une contrainte, c'est-à-dire des relations mathématiques entre plusieurs termes. Les types de relations présentés dans cette section correspondent aux opérateurs définis à la section 5.1.2.

Le catalogue présenté à cette section n'utilise pas intégralement le thésaurus visuel de Burns et Hajdukiewicz (2004), mais s'en inspire grandement. Ce dernier présente plusieurs manières de représenter graphiquement différentes relations entre des termes. Le présent catalogue se limite qu'à une seule manière par type de relation, c'est-à-dire par opérateur. Pour chaque opérateur, son composant visuel associé, sa représentation avec WoDoMoLEID et l'extrait du méta-modèle correspondant à cette dernière sont présentés.

Il est à noter que les composants visuels présentés dans cette section peuvent être modifiés ou remplacés. L'important est que leur structure corresponde à celle imposée par WoDoMoLEID. Le choix de leur forme est inspiré de Burns et Hajdukiewicz (2004), mais n'a pas fait l'objet de validation auprès de sujets humains.

6.2.1.1. Agrégation

Les trois opérateurs d'agrégation décrits à la section 5.1.2 sont : `Max`, `Min` et `Somme`. Leur composant visuel est illustré respectivement aux figures 6.6, 6.7 et 6.8. Un opérateur d'agrégation consiste à réaliser une opération sur un ensemble de valeurs qui

correspondent à un même terme, mais pour des objets différents. Un terme est représenté par une barre verticale grise où la valeur du terme est associée à sa hauteur.

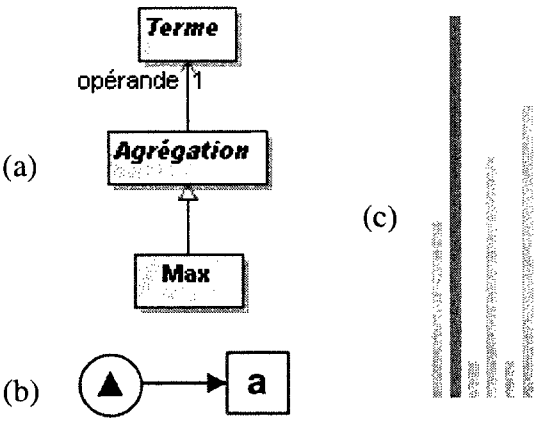


Figure 6.6 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `Max`.

En ce qui concerne les opérateurs `Max` et `Min`, le résultat de ces opérations consistent à identifier la valeur maximale ou minimale d'un ensemble de valeurs correspondant au terme `a`. Il est donc nécessaire de présenter chaque valeur de l'ensemble. Le résultat est identifié à l'aide d'une nuance différente, c'est-a-dire un gris plus foncé. À la figure 6.6, la barre grise foncée identifie la valeur maximale de l'ensemble de six valeurs. À la figure 6.7, sur les six valeurs, deux correspondent à la valeur minimale de l'ensemble. Elles sont toutes deux identifiés à l'aide d'un gris plus foncé. Si les valeurs de l'ensemble sont modifiées, la valeur maximale ou minimale est toujours identifiée en foncé.

La figure 6.8 illustre le composant visuel de l'opérateur `Somme`. Le résultat correspond à l'addition de toutes les valeurs de l'ensemble. Chaque valeur correspond à une section de la barre. L'exemple du composant visuel illustre le résultat de l'addition de quatre valeurs du terme `a`. Ainsi, si les valeurs de l'ensemble sont respectivement 1, 4, 3 et 2, le

résultat est 10. La hauteur de chaque section de la barre doit respecter le ratio de sa valeur par rapport au résultat de la somme. Si une des valeurs de l'ensemble est modifiée, la taille de la barre est modifiée en conséquence.

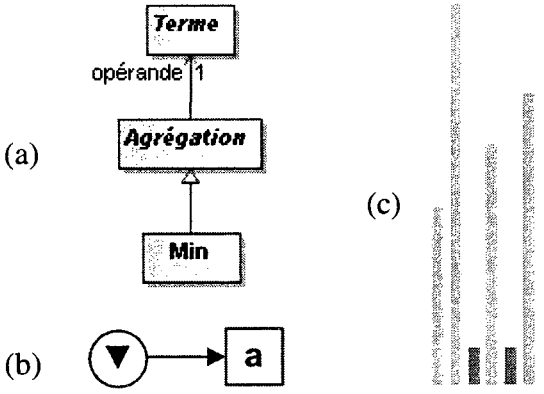


Figure 6.7 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `Min`.

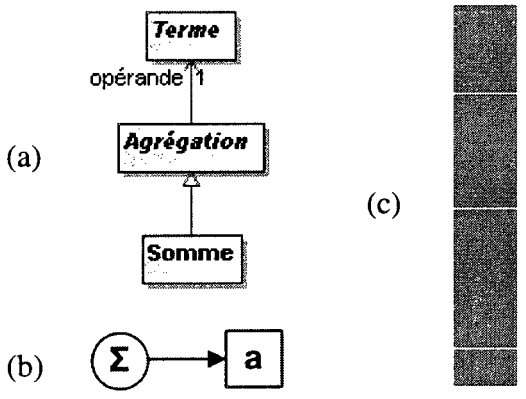


Figure 6.8 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `Somme`.

6.2.1.2. Arithmétique

Les quatre opérateurs arithmétiques décrits à la section 5.1.2 sont : `Addition`, `Soustraction`, `Multiplication` et `Division`.

La figure 6.9 illustre le composant visuel de l'opérateur `Addition`. Il s'agit de la combinaison de la valeur de deux termes. Le composant visuel est une barre à deux sections; chaque section représente la valeur d'un des deux termes. Si la valeur d'un des deux termes est modifiée, la taille de la barre est modifiée en conséquence.

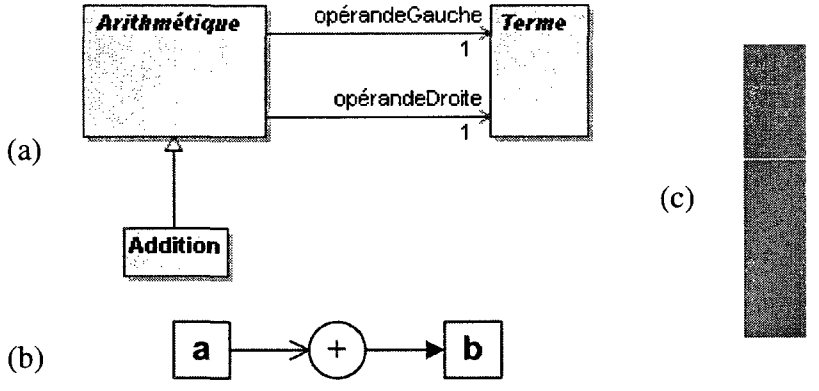


Figure 6.9 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `Addition`.

La figure 6.10 illustre le composant visuel de l'opérateur `Soustraction`. Cet opérateur porte également sur la relation entre deux termes. Deux exemples du composant visuel sont illustrés à la figure 6.10. L'exemple de gauche illustre la situation lorsque la valeur du terme `a` est supérieure à la valeur du terme `b`. La barre grise foncée représente la valeur du terme `a` et la barre grise pâle représente la valeur du terme `b`. Le résultat de l'opération est représenté par la section foncée inférieure. Il faut lire cette barre comme étant une barre grise foncée d'une certaine hauteur (valeur du terme `a`) à laquelle on retranche une barre grise pâle d'une certaine hauteur (valeur du terme `b`).

L'exemple de droite illustre la situation inverse, c'est-à-dire que la valeur du terme `a` est inférieure à la valeur du terme `b`. La barre grise foncée représente la valeur du terme `a` et la barre grise pâle représente la valeur du terme `b`. La section pâle inférieure représente

le résultat de l'opération. Toutefois, puisque la section pâle est plus grande que la section foncée, il faut interpréter le résultat de l'opération comme étant négatif.

Dans le cas où les valeurs respectives des termes `a` et `b` sont égale, la barre aura une seule section gris foncée.

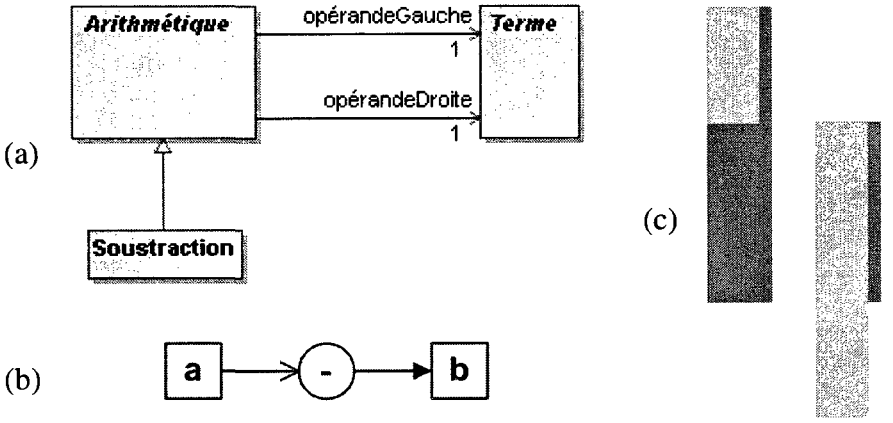


Figure 6.10 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur `Soustraction`.

La figure 6.11 illustre le composant visuel de l'opérateur `Multiplication`. Tout comme les opérations d'addition et de soustraction, la multiplication met en relation deux termes. Sa représentation géométrique est toutefois plus complexe. Deux exemples sont illustrés à la figure 6.11. Il s'agit d'une combinaison de trois barres. Les deux barres pâles représentent les valeurs des termes `a` (barre verticale) et `b` (barre horizontale). La barre foncée représente le résultat de l'opération. Les trois barres sont reliées par deux lignes, une horizontale et une diagonale. La taille de la diagonale peut changer, mais celle de la ligne horizontale reste fixe. L'intersection entre ces deux lignes est représentée par un point noir. Ce point est un pivot qui représente le centre de la relation de multiplication. Par exemple, si la valeur du terme `a` augmente, la ligne horizontale et le point prendront une position plus élevée. Ceci aura pour effet de déplacer la diagonale

de telle manière à positionner son extrémité supérieure gauche plus haut entraînant avec elle une augmentation de la taille de la barre foncée. Les deux exemples du composant visuels de la figure 6.11 illustrent un tel changement.

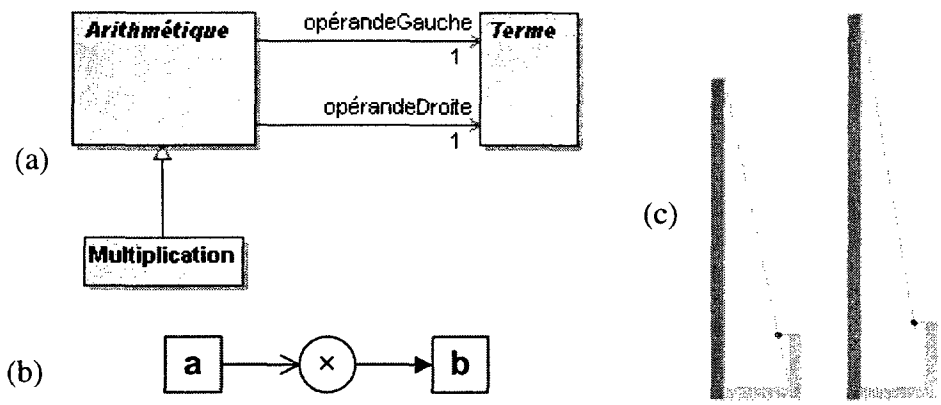


Figure 6.11 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur `Multiplication`.

La figure 6.12 illustre le composant visuel de l'opérateur `Division`. Ce composant visuel doit être interprété de la même manière que pour le composant visuel de l'opérateur `Multiplication`. Encore une fois, les barres pâles représentent les valeurs des termes `a` (barre verticale) et `b` (barre horizontale) et la barre foncée représente le résultat de l'opération.

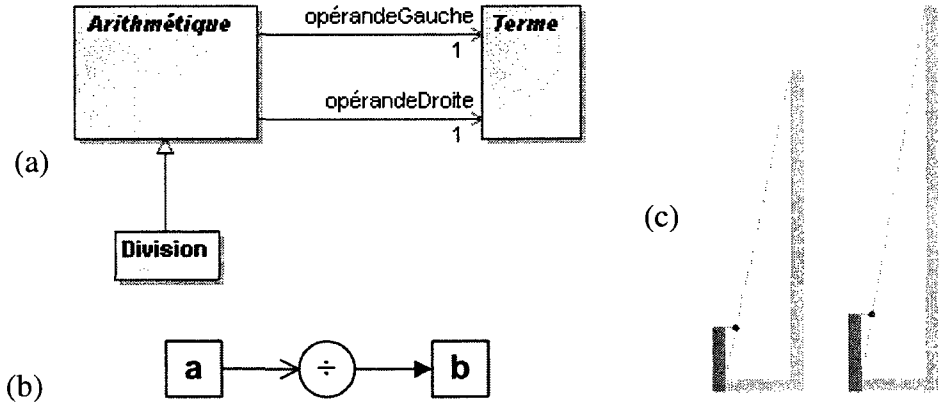


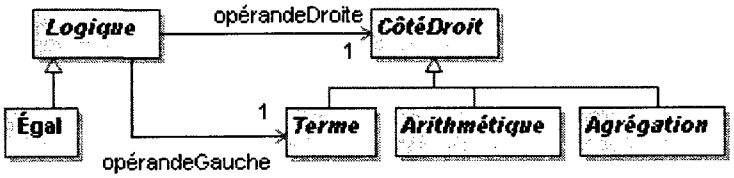
Figure 6.12 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur `Division`.

6.2.1.3. Logique

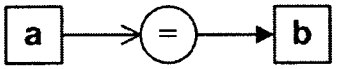
Les six opérateurs logiques décrits à la section 5.1.2 sont : `Égal`, `PasÉgal`, `PlusGrand`, `PlusGrandÉgal`, `PlusPetit` et `PlusPetitÉgal`.

La figure 6.13 illustre le composant visuel de l'opérateur `Égal`. Trois exemples du composant visuel sont illustrés avec des termes pour l'opérande de gauche (barre de gauche) et l'opérande de droite (barre de droite). La barre de gauche représente la valeur du terme `a` qui fait office de référence. Une ligne horizontale permet de comparer la valeur du terme `b`, la barre de droite, à cette référence. L'exemple du centre illustre la situation lorsque le résultat de l'opération est « vrai », c'est-à-dire lorsque les valeurs des termes `a` et `b` sont égales. Les deux autres exemples illustrent la situation lorsque le résultat de l'opération est « faux ». La couleur noire est utilisée pour représenter soit le manque à gagner, soit l'excédent.

(a)



(b)



(c)

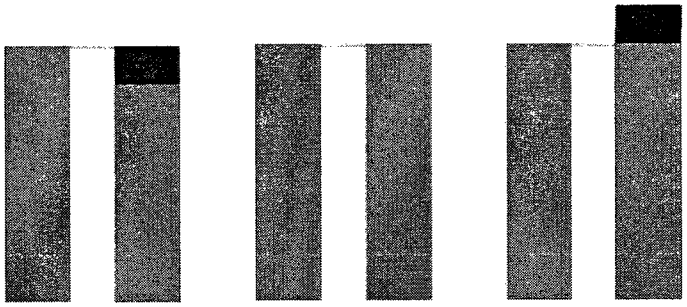


Figure 6.13 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `Égal`.

La figure 6.14 illustre le composant visuel de l'opérateur `PasÉgal`. Seul l'exemple du centre présente un résultat d'opération « faux ». Il s'agit de la situation où les valeurs des termes `a` et `b` sont égales; il y a donc un léger manque à gagner ou excédent représenté par la ligne noire en haut de la barre de droite. En ce qui concerne les deux autres exemples, ils représentent deux situations d'inégalités entre les valeurs des termes `a` et `b`; l'absence de couleur noire indique que le résultat de l'opération est « vrai ».

La figure 6.15 illustre le composant visuel de l'opérateur `PlusGrand`. Seul l'exemple de gauche indique un résultat « vrai » de l'opération. La couleur noire dans les deux autres exemples indiquent que la valeur du terme `a` n'est pas supérieure à la valeur du terme `b`.

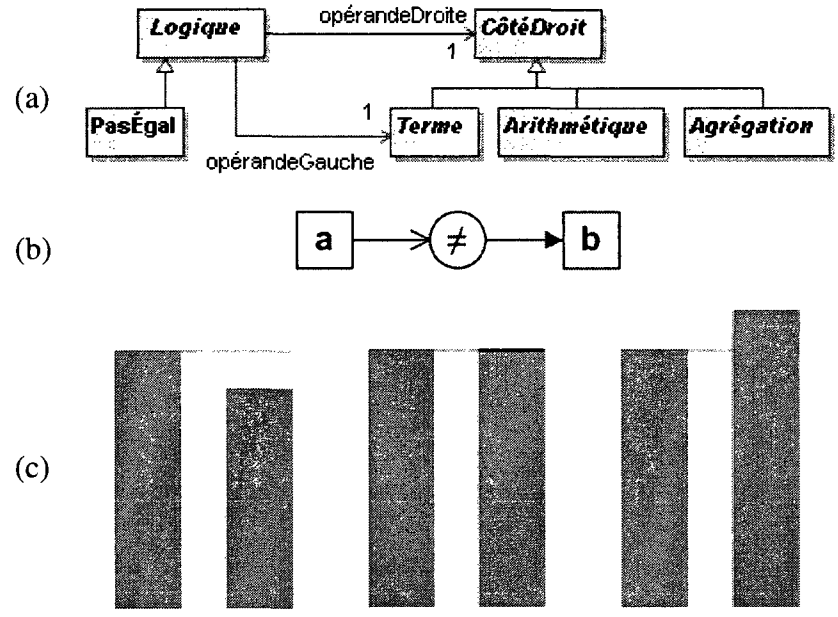


Figure 6.14 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `PasÉgal`.

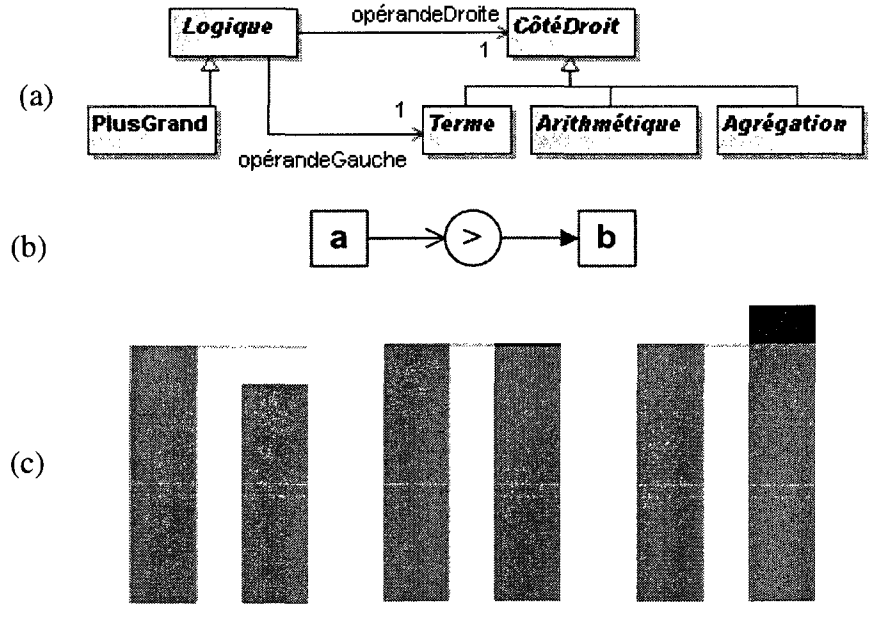


Figure 6.15 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `PlusGrand`.

La figure 6.16 illustre le composant visuel de l'opérateur `PlusGrandÉgal`. Dans ce cas, seul l'exemple de droite illustre une situation où le résultat de l'opération est « faux »; la valeur du terme `b` est supérieure à la valeur du terme `a`. Pour les deux autres exemples, la valeur est respectivement inférieure et égale.

Les figures 6.17 et 6.18 illustrent respectivement le composant visuel de l'opérateur `PlusPetit` et le composant visuel de l'opérateur `PlusPetitÉgal`. Les exemples suivent la même logique que celle des composants visuels des opérateurs `PlusGrand` et `PlusGrandÉgal`.

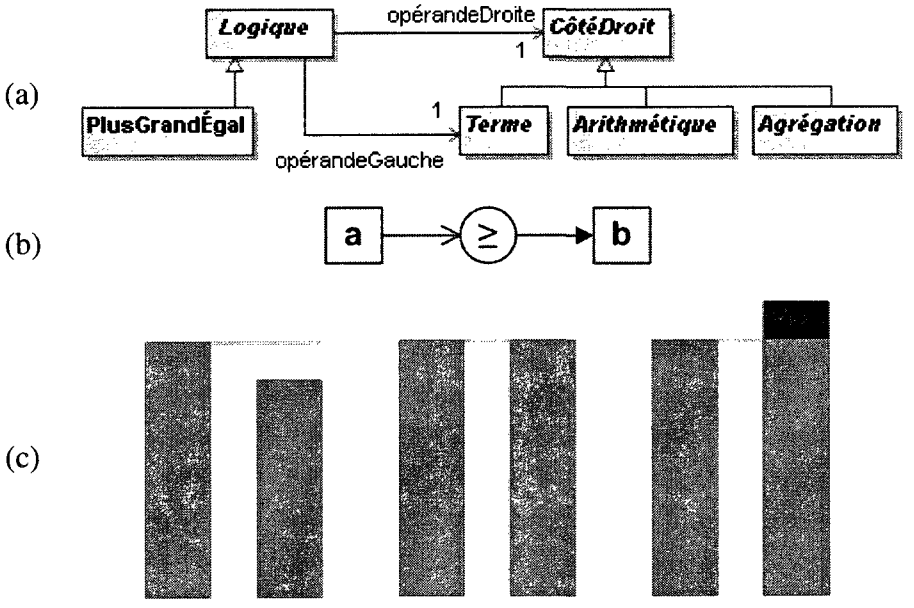


Figure 6.16 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `PlusGrandÉgal`.

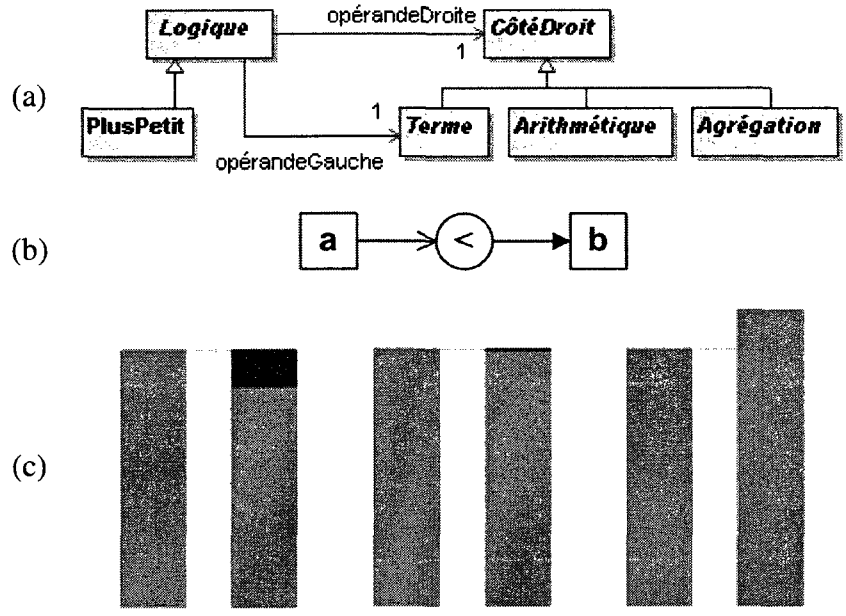


Figure 6.17 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `PlusPetit`.

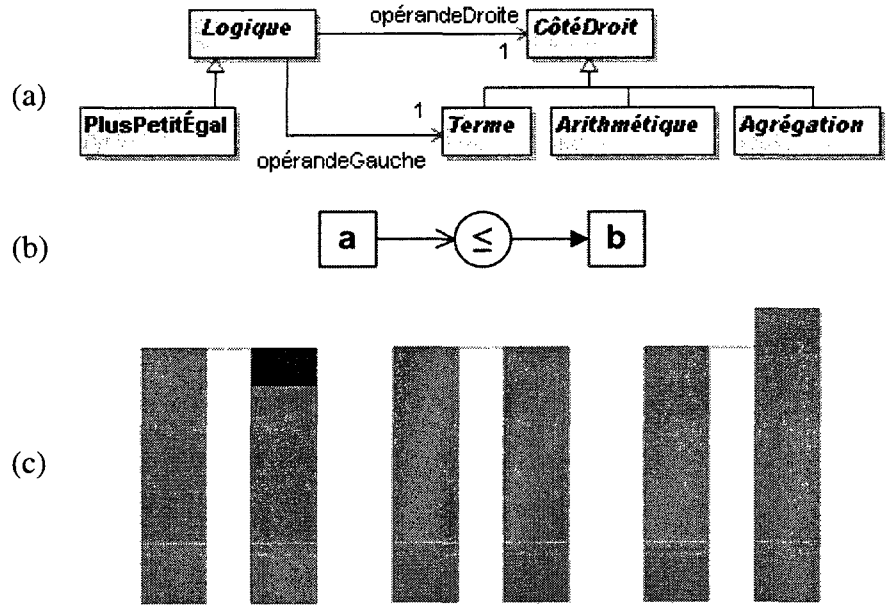
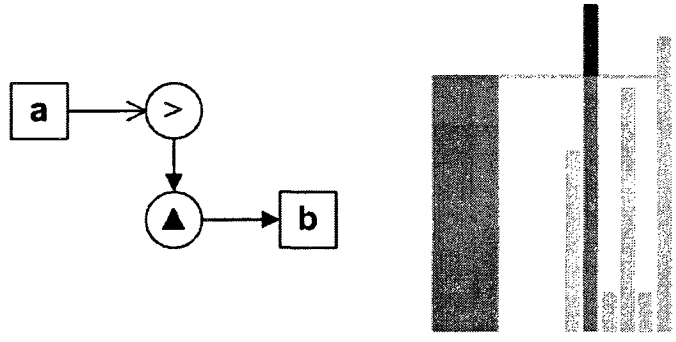


Figure 6.18 – (a) Extrait du méta-modèle, (b) représentation avec WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur `PlusPetitÉgal`.

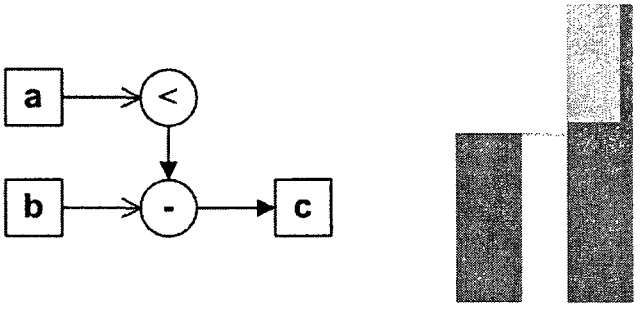
Les exemples précédents de composants visuels pour les opérateurs logiques utilisent tous le terme comme opérande de droite. Toutefois, tel que défini par le paquetage des opérateurs logiques du méta-modèle de WoDoMoLEID à la section 5.1.2.3, l'opérande de gauche peut être, en plus du terme, une opération d'agrégation ou une opération arithmétique.

Les figures 6.19 et 6.20 illustrent deux exemples d'opération logique avec, respectivement, une opération d'agrégation et une opération arithmétique. La figure 6.19 illustre un opérateur `PlusGrand` avec comme opérande de droite un opérateur `Max`. L'exemple illustre la situation où la valeur du terme `a` n'est pas supérieure au résultat de l'opérateur `Max`. La barre noire représente l'excédent. En ce qui concerne la figure 6.20, elle illustre un opérateur `PlusPetit` avec comme opérande de droite un opérateur `Soustraction`. L'exemple illustre la situation où la valeur du terme `a` est inférieure au résultat de l'opérateur `Soustraction`.



(a) (b)

Figure 6.19 – (a) Représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `PlusGrand` avec un opérateur `Max` comme opérande de droite.



(a) (b)

Figure 6.20 – (a) Représentation avec WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur `PlusPetit` avec un opérateur `Soustraction` comme opérande de droite.

6.2.2. Application à la gestion de portefeuille d'actifs financiers

La section précédente présente un catalogue de composants visuels réutilisables pour la CIE. Chaque composant visuel représente un opérateur et ses éléments associés (terme ou opérateur). Cette section vise à appliquer ce catalogue de composants visuels à un cas, celui de la gestion de portefeuille d'actifs financiers présenté à la section 6.1.2. Ce cas comporte onze contraintes réparties à traves cinq niveaux d'abstraction.

L'application au cas est réalisée en deux parties. La première partie porte sur la sélection des composants. Il s'agit de démontrer que WoDoMoLEID permet une correspondance directe entre les contraintes et les composants visuels. La seconde partie porte sur l'intégration des composants visuels en une IE. Il s'agit de démontrer que les composants sélectionnés peuvent être combinés pour constituer une IE.

6.2.2.1. Sélection des composants visuels

Cette section vise à associer chaque contrainte du cas de la gestion de portefeuille d'actifs financiers définie à la section 6.1.2 à son composant visuel correspondant. La figure 6.21 présente les onze contraintes numérotées et regroupées par niveau

d'abstraction. Les figures 6.22 à 6.32 présentent les composants visuels pour chacune de ces contraintes.

Niveau d'abstraction	No	Contrainte
1	1	Client.toléranceAuRisque ≥ Portefeuille.risque
	2	Client.objectifDeRendement ≤ Portefeuille.rendement
2	3	Portefeuille.rendement = Σ ActifFinancier.rendementPondéré
	4	Portefeuille.risque = Σ ActifFinancier.risquePondéré
3	5	ActifFinancier.rendementPondéré = ActifFinancier.rendement x ActifFinancier.pondération
	6	ActifFinancier.risquePondéré = ActifFinancier. risque x ActifFinancier.pondération
4	7	ActifFinancier.pondération = ActifFinancier.valeurTotale ÷ Portefeuille.valeurTotale
	8	ActifFinancier.rendement = ActifFinancier.différenceValeur ÷ ActifFinancier.valeurInitiale
5	9	ActifFinancier.différenceValeur = ActifFinancier.valeurActuelle - ActifFinancier.valeurInitiale
	10	Portefeuille.valeurTotale = Σ ActifFinancier.valeurTotale
	11	ActifFinancier. valeurTotale = ActifFinancier.valeurActuelle x ActifFinancier.quantité

Figure 6.21 – Les onze contraintes du domaine de travail.

La figure 6.22 illustre le composant visuel de l'opérateur `PlusGrandÉgal` avec deux termes comme opérandes de gauche et de droite.

(a) `Client.toléranceAuRisque ≥ Portefeuille.risque`

(b)

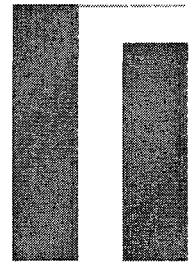


Figure 6.22 – (a) La contrainte 1 et (b) son composant visuel.

La figure 6.23 illustre le composant visuel de l'opérateur `PlusPetitÉgal` avec deux termes comme opérandes de gauche et de droite.

(a) `Client.objectifDeRendement ≤ Portefeuille.rendement`

(b)

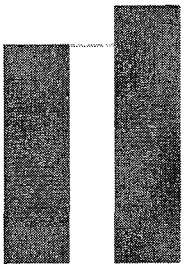


Figure 6.23 – (a) La contrainte 2 et (b) son composant visuel.

La figure 6.24 illustre le composant visuel de l'opérateur Égal avec un terme comme opérande de gauche et l'opérateur Somme comme opérande de droite.

(a) `Portefeuille.rendement = Σ ActifFinancier.rendementPondéré`

(b)

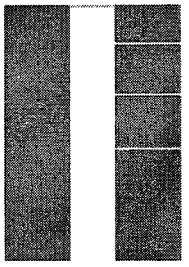


Figure 6.24 – (a) La contrainte 3 et (b) son composant visuel.

La figure 6.25 illustre le composant visuel de l'opérateur Égal avec un terme comme opérande de gauche et l'opérateur Somme comme opérande de droite.

(a) `Portefeuille.risque = Σ ActifFinancier.risquePondéré`

(b)

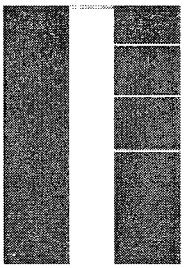


Figure 6.25 – (a) La contrainte 4 et (b) son composant visuel.

La figure 6.26 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)
```
ActifFinancier.rendementPondéré =
ActifFinancier.rendement x ActifFinancier.pondération
```

(b)

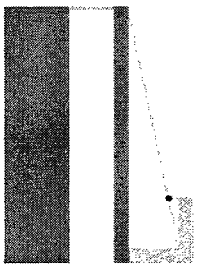


Figure 6.26 – (a) La contrainte 5 et (b) son composant visuel.

La figure 6.27 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)

```
ActifFinancier.risquePondéré =
ActifFinancier.risque x ActifFinancier.pondération
```

(b)

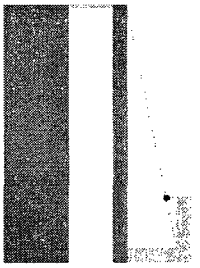


Figure 6.27 – (a) La contrainte 6 et (b) son composant visuel.

La figure 6.28 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Division` comme opérande de droite.

(a)

```
ActifFinancier.pondération =
ActifFinancier.valeurTotale ÷ Portefeuille.valeurTotale
```

(b)

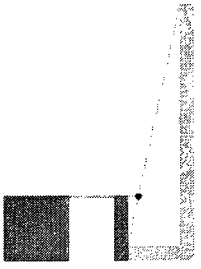


Figure 6.28 – (a) La contrainte 7 et (b) son composant visuel.

La figure 6.29 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Division` comme opérande de droite.

(a)
```
ActifFinancier.rendement =
ActifFinancier.différenceValeur ÷ ActifFinancier.valeurInitiale
```

(b)

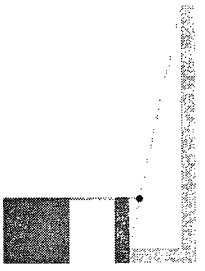


Figure 6.29 – (a) La contrainte 8 et (b) son composant visuel.

La figure 6.30 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Soustraction` comme opérande de droite.

(a)
```
ActifFinancier.différenceValeur =
ActifFinancier.valeurActuelle - ActifFinancier.valeurInitiale
```

(b)

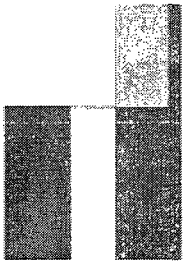


Figure 6.30 – (a) La contrainte 9 et (b) son composant visuel.

La figure 6.31 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Somme` comme opérande de droite.

(a) `Portefeuille.valeurTotale = Σ ActifFinancier.valeurTotale`

(b)

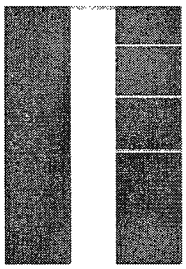


Figure 6.31 – (a) La contrainte 10 et (b) son composant visuel.

La figure 6.32 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)
```
ActifFinancier. valeurTotale =
ActifFinancier.valeurActuelle x ActifFinancier.quantité
```

(b)

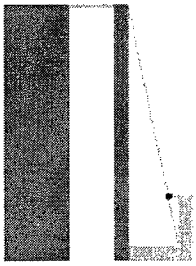


Figure 6.32 – (a) La contrainte 11 et (b) son composant visuel.

6.2.2.2. Intégration des composants visuels en une maquette d'interface écologique

La figure 6.33 illustre une maquette d'IE pour le cas de gestion de portefeuille d'actifs financiers. Cette maquette intègre les composants visuels sélectionnés à la section précédente. Il est à noter que les composants visuels ne représentent pas de valeurs cohérentes; il s'agit uniquement d'illustrer leur intégration en une IE en les disposant sur une surface d'affichage.

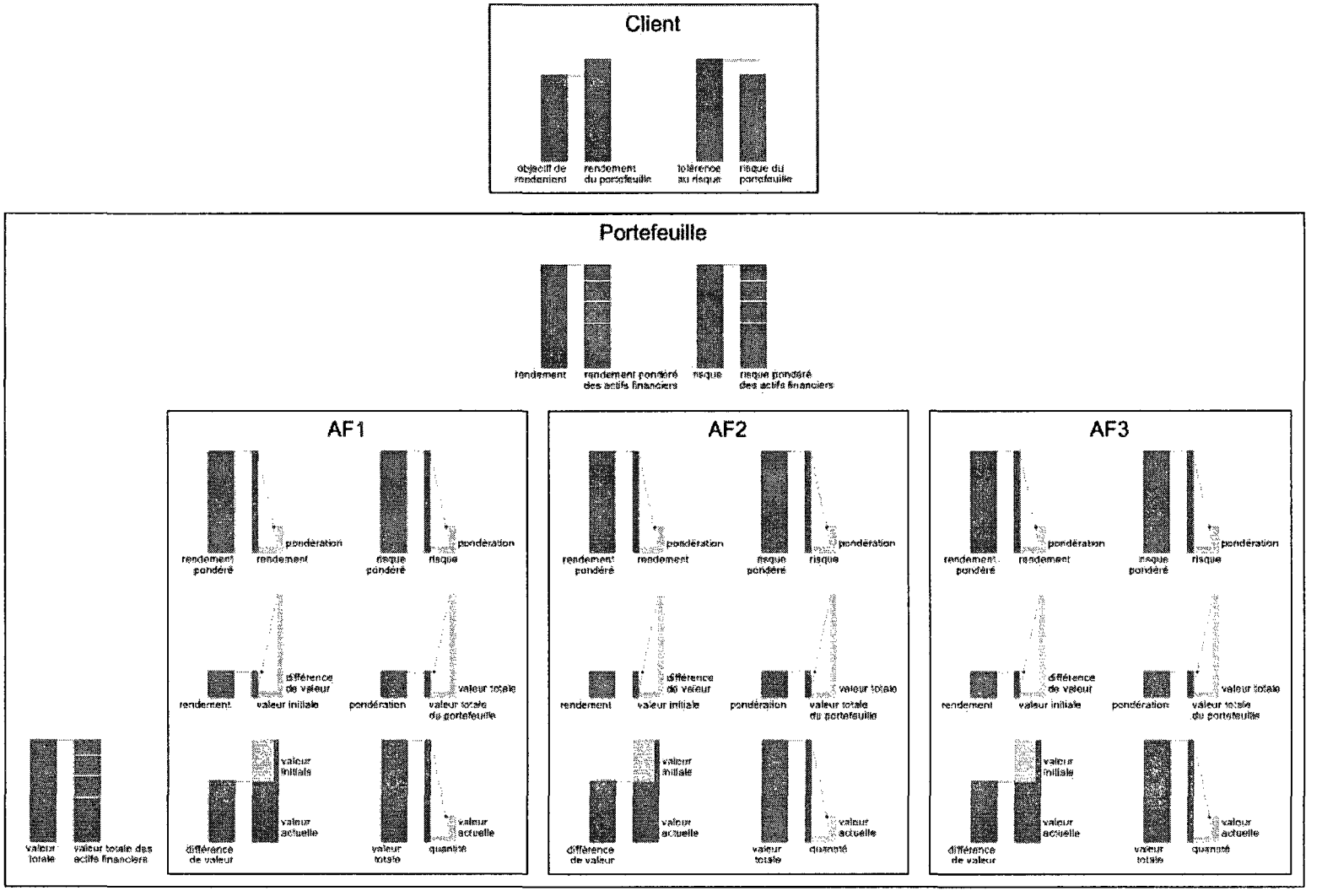


Figure 6.33 – Maquette d'une interface écologique pour la gestion de portefeuille d'actifs financiers.

Cette IE présente un portefeuille contenant trois actifs financiers (AF1, AF2 et AF3). Les mêmes composants visuels sont réutilisés pour représenter les contraintes associées à chaque actif financier. Les composants visuels sont disposés sur cinq rangées. Chaque rangée correspond à un des cinq niveaux d'abstraction identifié à la figure 6.21; les composants visuels de la première rangée du haut représentent les contraintes du premier niveau d'abstraction et les composants visuels de la dernière rangée du bas représentent les contraintes du cinquième niveau d'abstraction.

6.2.3. Discussion

Les sections 6.2.2.1 et 6.2.2.2 démontrent la relation directe entre les contraintes et les composants visuels. À la section 6.2.2.1, un catalogue de composants visuels réutilisables est présenté. La structure de chaque composant visuel correspond à la structure d'un opérateur particulier de WoDoMoLEID. Ainsi, lorsque le domaine de travail est modélisé à l'aide de WoDoMoLEID, il est possible d'associer directement un composant visuel particulier à un type d'opérateur, par exemple une addition ou une égalité.

Cette affirmation est vérifiée à la section 6.2.2.2 dans laquelle des composants visuels ont été sélectionnés afin de créer une IE pour représenter un portefeuille d'actifs financiers. Cette application démontre la réutilisation des composants visuels. En effet, le cas comporte onze contraintes réparties en cinq niveaux d'abstraction. Plusieurs de ces contraintes s'appuient sur les mêmes opérateurs, par exemple les contraintes 3, 4 et 10 avec un opérateur `Égal` et un opérateur `Somme` ainsi que les contraintes 7 et 8 avec un opérateur `Égal` et un opérateur `Division`. Même si elles s'appuient sur des termes différents, ces contraintes sont associées aux mêmes composants visuels puisqu'elles utilisent les mêmes opérateurs. La raison est que la définition des contraintes du

domaine de travail avec WoDoMoLEID s'appuie sur la même structure que celle sous-jacente aux composants visuels.

Cette section a démontré que WoDoMoLEID permet une correspondance directe entre la définition des contraintes d'un domaine de travail et les composants visuels. Établir cette correspondance est le premier pas vers l'automatisation de la démarche de sélection des composants visuels. Toutefois, le travail de conception ne se limite pas à cette tâche. Le concepteur doit principalement s'acquitter de deux autres tâches : (a) la création des composants visuels et (b) la disposition de ceux-ci sur la surface d'affichage. Ces deux tâches requièrent l'application de règles ergonomiques de présentation d'informations[26]. Ces règles ne sont pas propres aux IE, mais aux IU en général. Puisque cette thèse porte exclusivement sur la CIE, les considérations externes à ce type d'IU sont omises.

[26] Pour la littérature à ce sujet, voir entre autres Tullis (1997), Sanders et McCormick (1993) ainsi que Mayhew (1992).

CHAPITRE 7 - CONCLUSION

Cette thèse présente un nouveau langage de modélisation de domaines de travail quantitatifs pour la CIE appelé WoDoMoLEID. Il peut être considéré comme un remplacement de la HAD ou il peut être utilisé conjointement avec cette technique de représentation de domaines de travail. L'objectif de ce nouveau langage est de faciliter le travail du concepteur d'IE en permettant une correspondance directe entre la définition des contraintes du domaine de travail et les composants visuels qui les représentent et qui composent l'IE.

Ce chapitre présente les contributions de cette thèse à la section 7.1. Les limites sont présentées à la section 7.2. Dernièrement, la section 7.3 présente les avenues futures de recherche.

7.1. Contributions

Plusieurs contributions sont issues de cette thèse. Premièrement, les chapitres 3 et 4 présentent, respectivement, un méta-modèle des IE et un méta-modèle de la HAD. Ces méta-modèles apportent une compréhension plus précise de ces techniques de représentation en formalisant le vocabulaire et la grammaire de ces derniers.

Deuxièmement, le chapitre 4 présente une évaluation de la compatibilité entre la HAD et les IE. La littérature fait état de deux problèmes avec la HAD. Le premier porte sur sa sémantique imprécise et le deuxième porte sur son insuffisance pour la CIE. Cette évaluation permet de constater que la nature de ces deux problèmes est liée à la structure de la HAD en tant que technique de représentation.

Troisièmement, les résultats de l'évaluation réalisée au chapitre 4 s'appuie sur une méthode inédite d'évaluation de la compatibilité entre deux méta-modèles. Cette méthode propose l'utilisation d'un taux de correspondance qui est obtenu suite à une évaluation qualitative de la correspondance entre les éléments de chaque méta-modèle.

Quatrièmement, un nouveau langage de modélisation de domaines de travail, appelé WoDoMoLEID, est proposé. Ce langage, de par sa structure, apporte une solution aux deux problèmes de la HAD. Premièrement, il tient compte, de manière intégrée, de tous les éléments d'un domaine de travail nécessaires à la CIE (objets, contraintes, hiérarchie fonctionnelle). Un symbole différent est associé à chaque élément du méta-modèle éliminant ainsi toute ambigüité dans l'interprétation de la syntaxe concrète. Deuxièmement, le langage proposé impose une structure dans la manière de définir les contraintes. Cette structure, qui dicte la manière dont les opérateurs et les termes sont reliés, permet une correspondance directe entre les contraintes et les composants visuels qui composent une IE. Cette correspondance directe est un premier pas dans l'automatisation de la sélection de composants visuels réutilisables.

Cinquièmement, un prototype de logiciel de modélisation a été développé. Ce dernier implémente la syntaxe abstraite et la syntaxe concrète de WoDoMoLEID. Ainsi, ce logiciel de modélisation renforce les règles du langage imposées par sa syntaxe abstraite. Il permet d'indiquer à son utilisateur qu'un modèle est bien formé ou non. Également, à partir d'un modèle du domaine, un algorithme permet de dériver automatiquement la hiérarchie fonctionnelle en associant un niveau de « profondeur » à chaque contrainte. L'utilisateur peut ainsi se concentrer sur la définition des contraintes du domaine de travail sans avoir à se soucier de les positionner dans la hiérarchie fonctionnelle.

Sixièmement, un catalogue de composants visuels réutilisables est proposé. Ce catalogue regroupe un ensemble de composants visuels dont chacun correspond à la représentation graphique d'un type d'opérateur et les relations avec ses opérandes. Ce catalogue évite de devoir concevoir les composants visuels à chaque nouvelle IE; le concepteur peut les sélectionner et les disposer sur la surface d'affichage. Il en résulte une diminution du délai du processus de conception d'IE.

7.2. Les limites

Bien que cette thèse présente plusieurs contributions, elle présente également certaines limites. Premièrement, la méthode d'évaluation de la compatibilité présentée au chapitre 4 s'appuie uniquement sur une évaluation de la correspondance entre les éléments des méta-modèles; les associations entre ces éléments ne sont pas considérées. Par conséquent, une évaluation pourrait avoir un taux de compatibilité de 100%, mais avoir des associations différentes entre les éléments des méta-modèles respectifs. Dans ce cas, il est incorrect de conclure que ces méta-modèles sont compatibles.

Deuxièmement, le degré de complexité du cas employé au chapitre 6 pour l'évaluation du langage, la gestion de portefeuille d'actifs financiers, a été déterminé afin de servir l'objectif d'évaluation uniquement. Ce cas est représentatif de la réalité en ce sens que les contraintes définies sont celles réellement utilisées dans ce domaine. Néanmoins, il est raisonnable de se questionner sur la modélisation d'un domaine de travail comprenant un plus grand nombre de contraintes. Est-ce que la syntaxe concrète telle que définie à la section 5.2 crée un obstacle à la lisibilité du modèle pour un domaine plus complexe, c'est-à-dire qui comporte plus de concepts, de termes et d'opérateurs que le cas de la gestion de portefeuille d'actifs financiers? Rien dans cette thèse ne permet de répondre à cette question.

Troisièmement, le langage de modélisation proposé permet de représenter des domaines de travail de nature quantitative seulement; il ne tient pas compte de la modélisation de domaines de travail qualitatifs. Burns et Hajdukiewicz (2004) ainsi que Rasmussen, Pjetersen et Goodstein (1994) présentent quelques domaines de travail de nature qualitative. Il est impossible de les modéliser avec WoDoMoLEID dans son état actuel.

Quatrièmement, la correspondance directe entre la structure des contraintes telles que définies par WoDoMoLEID et la structure des composants visuels présentées à la section 6.2.1 est un premier pas vers l'automatisation du processus de sélection des composants visuels. Toutefois, cette automatisation n'a pas été réalisée.

7.3. Avenues futures de recherche

Cette section présente quelques avenues de recherches pouvant être empruntées suite à cette thèse. Certaines de ces avenues sont associées aux limites identifiées à la section précédente.

La première avenue de recherche porte sur l'amélioration de WoDoMoLEID. Cette avenue se divise en trois voies. La première voie consiste à modifier la syntaxe abstraite de manière à ajouter des mécanismes de représentation de contraintes qualitatives. La deuxième voie consiste à modifier la syntaxe concrète pour la rendre plus utilisable. En effet, la syntaxe concrète présentée dans cette thèse a été créée arbitrairement sans savoir si les symboles utilisés pouvaient être interprétés correctement. Elle pourrait donc faire l'objet d'expériences avec des sujets humains. La troisième voie porte la modélisation de domaines très complexes, c'est-à-dire qui comporte un grand nombre de concepts, de termes et d'opérateurs. Afin de permettre la lisibilité des modèles, il serait nécessaire de permettre un mécanisme visuel permettant de scinder le modèle en plusieurs sous-

modèles. Une solution serait d'introduire la notion de paquetages utilisée par le langage UML (OMG 2008b).

La deuxième avenue de recherche, qui se divise également en trois voies, porte sur les composants visuels. La première voie consiste à enrichir le catalogue de composants visuels réutilisables. Pour ce faire, il faudrait proposer différents agencements de formes géométriques et les soumettre à des évaluations par des sujets humains. Précisément dans cette optique, Jessa et Burns (2007) ont réalisé plusieurs expériences visant à déterminer la sensibilité visuelle d'affichages graphiques dynamiques. En s'inspirant de leurs travaux, il serait intéressant d'évaluer l'impact de caractéristiques, par exemple la taille, la couleur et la disposition, sur la performance humaine. La deuxième voie porte sur l'automatisation du processus de sélection des composants visuels. Cette thèse présente les règles nécessaires à cet effet. Un prototype de logiciel pourrait être développé permettant de sélectionner les composants visuels appropriés à partir des contraintes définies avec WoDoMoLEID. Cette voie va de pair avec la troisième qui consiste à intégrer les règles ergonomiques nécessaires à l'optimisation de l'utilisabilité de l'IE. Ces règles qui serviraient à définir, entre autres, la disposition des composants visuels sur la surface d'affichage et les mécanismes de modification de valeurs pourraient être définies et ainsi être prises en considération pour l'automatisation du processus de conception de l'IE.

La troisième avenue de recherche porte sur les logiciels de modélisation. Le prototype présenté à la section 6.1.1 a été créé dans le but de mettre le langage en application. Les zones ont été créées arbitrairement et seules les fonctionnalités de base (ouvrir, sauvegarder, manipuler et valider un modèle) ont été implémentées. Un nouveau prototype pourrait être développé en offrant d'autres fonctionnalités comme la possibilité de voir le modèle sous forme d'arborescence ou de visionner un sous-

ensemble du modèle. Surtout, il est important que le prototype de logiciel de modélisation soit conçu de manière ergonomique.

La quatrième avenue de recherche porte sur la méthode d'évaluation de la compatibilité de deux méta-modèles. Un méta-modèle présente le vocabulaire et la grammaire d'un langage de modélisation. Ainsi, ces deux éléments doivent être pris en compte dans l'évaluation de la compatibilité. Il est donc nécessaire d'intégrer la prise en compte des associations dans le calcul du taux de correspondance.

Vicente (2002) présente un certain nombre de facteurs qu'il considère comme faisant obstacle à l'utilisation d'IE dans l'industrie. Un de ceux-ci est associé à la complexité et la lourdeur de l'approche de conception. Ces avenues de recherche permettront, espérons-le, d'alléger ce processus en automatisant la plupart des étapes entre l'analyse du domaine de travail et l'IE finale sous forme de code source. Ainsi, il sera plus facile de concevoir et réaliser des environnements de travail qui soutiendront davantage le processus de résolution de problème d'un individu, particulièrement lorsque celui-ci est aux prises avec un événement imprévu.

L'accroissement de la complexité des systèmes va de paire avec l'aggravation des conséquences en cas de désastre. Par conséquent, soutenir l'individu responsable du contrôle de ces systèmes lors de situations imprévues n'est plus un besoin, mais une nécessité.

RÉFÉRENCES

ACHONU, Jackie, JAMIESON, Greg (2003). Work domain analysis of a financial system: An abstraction hierarchy for portfolio management. *Proceedings of the 22nd European Annual Conference on Human Decision Making and Control*, (pp. 103-109). Linköping, Sweden. Cognitive Systems Engineering Lab.

ALEXANDER, Christopher, ISHIKAWA, Sara, SILVERSTEIN, Murray, JACOBSON, Max, FIKSDAHL-KING, Ingrid, ANGEL, Shlomo (1977). *A pattern language: towns, buildings, construction.* New York: Oxford University Press.

BALASUBRMANIAN, Krishnakumar, GOKHALE, Anirrunddha S., KARSAI, Gabor, SZTIPANOVITS, Janos, NEEMA, Sandeep (2006). Developing applications using model-driven design environments. *IEEE Computer*, 39(2), 33-40.

BÉZIVIN, Jean, GERBÉ, Olivier (2001). Towards a precise definition of the OMG/MDA framework [Version électronique]. *Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE 2001), San Diego, California* (pp. 273-280). Los Alamitos, CA: IEEE.

BILLET, Hélène, MORINEAU, Thierry (2005). Application du cadre des interfaces écologiques au domaine de la stratégie financière [Version électronique]. *MajecSTIC 2005 : Manifestation des Jeunes Chercheurs francophones dans les domaines des STIC*, Rennes, France (pp. 308-315). INRIA.

BISANTZ, Ann M., VICENTE, Kim J. (1994). Making the abstraction decomposition concrete. *International Journal of Human-Computer Studies*, 40(1), 93-117.

BORCHERS, Jan (2001). *A Pattern Approach to Interaction Design*. New York: John Wiley & Sons, Inc.

BURNS, Catherine M. (2000). Putting it all together: improving display integration in ecological displays. *Human Factors*, 42(2), 226-241.

BURNS, Catherine M., HAJDUKIEWICZ, John R. (2004). *Ecological interface design*. Boca Raton: CRC Press.

BURNS, Catherine M., KUO, Johnson, NG, Sylvia (2003). Ecological interface design: a new approach for visualizing network management. *Computer Networks*, 43, 36-388.

CHEN, Peter P. (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 9-36.

CHEN, Peter P. (1999). From ancient Egyptian language to future conceptual modeling [Version électronique]. *Proceedings of Symposium on Conceptual Modeling: Current Issues and Future Directions* (pp. 57-66). Londres: Springer-Verlag.

DAINOFF, Marvin J., DAINOFF, Charles A., MCFEETERS, Larry (2004). On the application of cognitive work analysis to the development of a commercial investment software tool [Version électronique]. *Proceedings of the Human Factors and Ergonomics Society 48th annual meeting* (pp. 595-599). Santa Monica, CA: Human Factors and Ergonomics Society.

DIAPER, Dan (2004). Understanding task analysis for human-computer interaction. In Dan Diaper et Neville Stanton (éd.), *The Handbook of Task Analysis for Human-Computer Interaction* (pp. 5-48). Mahwah: Lawrence Erlbaum Associates inc.

DORST, Kees (2006). Design Problems and Design Paradoxes. *Design Issues*, 22(3). 4-17.

FAVRE, Jean-Marie (2004). Foundations of model (driven) (reverse) engineering: models – episode 1: stories of the fidus papyrus and of the solarus. *Language engineering for model-driven software development.* Dagstuhl: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI). Consulté le 9 février 2009, tiré de http://drops.dagstuhl.de/opus/volltexte/2005/13/

FENN, Jackie (2007). *Gartner's Hype Cycle Special Report for 2007: Hype Cycle for Application Development, 2007.* Consulté le 17 janvier 2008, tiré de Gartner: http://www.gartner.com

FRANKEL, David S. (2003). *Model driven architecture: applying MDA to enterprise computing.* Indianapolis: Wiley Publishing Inc.

GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, John M. (1995). *Design patterns: elements of reusable object-oriented software.* Reading: Addison-Wesley Professional.

GIBSON, James J. (1979). *The ecological approach to visual perception.* Hillsdale: Lawrence Erlbaum Associates, Inc.

GOEL, Vinod (1992). Comparison of well-structured & ill-structured task environments. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum. Consulté le 9 février 2009, tiré de Goel Cognitive Neuroscience Lab, http://www.yorku.ca/vgoel

GOODSTEIN, L. P. (1981). Discrimative display support for process operators. In Jens Rasmussen et William B. Rouse (éd.), *Human Detection and Diagnosis of Systems Failures* (pp. 433-439). New York: Plenum Press.

GRAHAM, Ian (2002). *A Pattern Language for Web Usability.* Reading: Addison-Wesley Professional.

HAJDUKIEWICZ, John R., VICENTE, Kim J. (2004). A theoretical note on the relationship between work domain analysis and task analysis. *Theoretical Issues in Ergonomics Science*, 5(6), 527-538.

HAJDUKIEWICZ, John R., VICENTE, Kim J., DOYLE, D.J., MILGRAM, Paul, BURNS, Catherine M. (2001). Modeling a medical environment: an ontology for integrated medical informatics design. *International Journal of Medical Informatics*, 62(1), 79-99.

HAM, D.-H., YOON, W. C. (2001). Design of information content and layout for process control based on goal-means domain analysis. *Cognition Technology & Work*, 3(4), 205-223.

HENNICKER, Rolf, KOCH, Nora (2001). Modeling the User Interface of Web Applications with UML. *Practical UML-Based Rigorous Development Methods - Countering or Integrating the* eXtremists, *Workshop of the pUML-Group at the UML 2001, Toronto, Canada* (pp. 158-172). Consulté le 23 mai 2006, tiré de Nora Koch, http://www.pst.ifi.lmu.de/people/staff/koch/

HUGHES, R. I. G. (1997). Models and representation. *Philosophy of Science*, 64(4, supplement), S325-S336.

JESSA, Munira, BURNS, Catherine M. (2007). Visual sensitivities of dynamic graphical displays. *International Journal of Human-Computer Studies*, 65(3), 206-222.

KELLY, Steve, TOLVANEN, Juha-Pekka (2008). *Domain-specific modeling: enabling full code generation.* Hoboken: John Wiley & Sons, Inc.

KENT, Stuart (2002). Model driven engineering. *Proceedings of the Third International Conference on Integrated Formal Methods (IFM 2002), Turku, Finlande* (pp. 286-298). Berlin: Springer. Consulté le 23 mai 2006, tiré de Springer, http://www.springerlink.com

KÜHNE, Thomas (2006). Matters of (meta-) modeling. *Software and Systems Modeling*, 5(4), 369-385.

KULKARNI, Vinay, REDDY, Sreedhar (2003). Separation of concerns in model-driven development. *IEEE Software*, 20(5), 64.-69.

KURTEV, Ivan (2005). *Adaptability of model transformations*. Ph.D. inédit. Université de Twente, Pays-Bas.

LARSSEN, Jan Eric (1996). Diagnosis based on explicit means-end models. *Artificial Intelligence*, 80(1), 29-93.

LIN, Y., ZHANG, W.J. (2004). Towards a novel interface design framework: function-behavior-state paradigm. *International Journal of Human Computer Studies*, 61, 259-297.

LIN, Y., ZHANG, W.J. (2005). A function-behavior-state approach to designing human-machine interface for nuclear power plant operators. *IEEE Transactions on Nuclear Science*, 52(1), 430-439.

LIND, Morten (1994). Modeling goals and functions of complex industrial plant. *Journal of Applied Artificial Intelligence*, 8(2), 259-283.

LIND, Morten (1999). Making sense of the abstraction hierarchy. *Proceedings of the Conference on Cogntive Sceince Approaches to Process Control (CSAPC'99), Villeneuve d'Asc, France* (pp. 195-200). Consulté le 24 juillet 2006, tiré de Morten Lind, http://www.iau.dtu.dk/~ml

LIND, Morten (2003). Making sense of the abstraction hierarchy in the power plant domain. *Cognition Technology & Work*, 5(3), 67-81.

LO GIUDICE, Diego (2007). *The State Of Model-Driven Development.* Consulté le 17 janvier 2008, tiré de Forrester: http://www.forrester.com

MAGUIRE, Martin (2001). Methods to support human-centered design. *International Journal of Human-Computer Studies*, 55, 587-634.

MARCH, James G., SIMON, Herbert A. (1958). *Organizations.* New York: John Wiley & Sons, Inc.

MAYHEW, Deborah J. (1992). Screen Layout and Design. In *Principles and guidelines in software user interface design* (pp. 458-506). Englewood Cliffs: Prentice-Hall.

MELLOR, Stephen J., CLARK, Anthony N., FUTAGAMI, Takao (2003). Model-driven development. *IEEE software*, 20(5), 14-18.

MELLOR, Stephen J., SCOTT, Kendall, UHL, Axel, WEISE, Dirk (2004). *MDA distilled: principles of model-driven architecture.* Reading: Addison-Wesley Professional.

MILLER, Anne (2000). Patient monitoring systems for effective patient management in the ICU: friend or foe? *Proceedings of the Joint Meeting of the Human Factors and Ergonomics Society and the International Ergonomics Association (IEA2000/HFES2000), Santa Monica, California* (pp. 4-262-4-265). Human Factors and Ergonomics Society. Consulté le 9 février 2009, tiré de The University of Queensland – Cognitive Engineering Research Group, http://www.itee.uq.edu.au/~cerg

MILLER, Christopher A., VICENTE, Kim J. (2001). Comparison of display requirements generated via hierarchical task analysis and abstraction-decomposition space analysis techniques. *International Journal of Cognitive Ergonomics*, 5(3), 335-355.

MILLER, George A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63(2), 81-97.

MOÏSE, Alexandre, NOISEUX, Marie Hélène (2007). *Un modèle d'informations intégré pour la gestion de portefeuille d'actifs financiers.* Chaire en management des services financiers, École des sciences de la gestion, Université du Québec à Montréal.

MOÏSE, Alexandre, ROBERT, Jean-Marc (2007a). Un langage de modélisation pour la présentation abstraite d'une interface écologique. *Actes du 7e Congrès international de génie industriel (CIGI2007), Trois-Rivières, Canada.*

MOÏSE, Alexandre, ROBERT, Jean-Marc (2007b). Model driven ecological interface development: the constraints model. *Proceedings of the 7th Workshop on Domain-Specific Modeling (DSM 2007), ooPSLA 2007, Montréal, Canada* (pp. 127-138). Jyväskylä: Jyväskylä University Printing House.

NEWELL, Alan, SIMON, Herbert A. (1972). *Human problem solving*. Englewood Cliffs: Prentice-Hall.

NORMAN, Donald A. (1993). *Things that make us smart.* Cambrige: Perseus Books.

NORMAN, Donald A. (2002). *The design of everyday things.* New York: Basic Books.

NUNES, Nuno J., CUHNA, Joao F. (2000). Wisdom: a software engineering method for small software development companies. *IEEE Software*, 17(5), 113-119.

OBJECT MANAGEMENT GROUP. (2006). *Object Constraint Language, version 2.0.* Object Management Group, formal/06-05-01.

OBJECT MANAGEMENT GROUP. (2008a). *OMG Unified Modeling Language (OMG UML), Infrastructure, version 2.2.* Object Management Group, ptc/08-05-04.

OBJECT MANAGEMENT GROUP. (2008b). *OMG Unified Modeling Language (OMG UML), Superstructure, version 2.2.* Object Management Group, ptc/08-05-05.

ODELL, James, RAMACKERS, Guus J. (1997). Toward a Formalization of 00 Analysis. *Journal of Object-Oriented Programming*, 10(4), 64-68.

PAIVIO, Allan (2007). *Mind and its evolution: a dual coding theoretical approach.* Mahwah: Lawrence Erlbaum Associates, inc.

PAWLAK, William S., VICENTE, Kim J. (1996). Inducing effective operator control through ecological interface design. *International Journal of Human-Computer Studies*, 44(5), 653-688.

PINHEIRO DA SILVA, Paulo, PATON, Norman W. (2003). User Interface Modeling in UML*i*. *IEEE software*, 20(4), 2-69.

RASMUSSEN, Jens (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, smc-13(3), 257-266.

RASMUSSEN, Jens (1985). The role of hierarchical knowledge representation in decisionmaking and system management. *IEEE Transactions on Systems, Man, and Cybernetics,* smc-15(2), 234-243.

RASMUSSEN, Jens, PEJTERSEN, Annelise Mark, GOODSTEIN, L. P. (1994). *Cognitive Systems Engineering*. New York: John Wiley & Sons, Inc.

RASMUSSEN, Jens, PEJTERSEN, Annelise Mark, SCHMIDT, Kjeld (1990). *Taxonomy for cognitive work analysis*. (Risø-M-2871). Roskilde: Risø National Laboratory.

REASON, James (1990). *Human error*. Cambridge: Cambridge University Press.

SANDERS, Mark S., MCCORMICK, Ernest James (1993). Visual displays of dynamic information. In Mark S. Sanders et Ernest James McCormick(éd.), *Human Factors in Engineering and Design, seventh edition* (pp. 132-159). New York: McGraw-Hill.

SCHMIDT, Douglas C. (2006). Model-driven engineering. *IEEE Computer*, 39(2), 25-31.

SEIDEWITZ, Ed (2003). What models mean. *IEEE Software*, 20(5), 64.-69.

SELIC, Bran (2003). The pragmatics of model-driven development. *IEEE Software*, 20(5), 19-25.

SHARP, Thomas D., HELMICKI, Arthur J. (1998). The application of the ecological interface design approach to neonatal intensive care medicine. *Proceedings of the 42nd Annual Meeting of the Human Factors and Ergonomics Society* (pp. 350-354). Santa Monica: Human Factors and Ergonomics Society. Consulté le 9 février 2009, tiré de SDL, http://www.sdltd.com

SHNEIDERMAN, Ben (1982). The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, 1(3), 237-256.

SIMON, Herbert A. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63(2), 129-138.

SIMON, Herbert A. (1973). The structure of ill-structured problems. *Artificial Intelligence*, 4(3-4), 181-201.

SIMON, Herbert A. (1997). *Administrative behavior, Fourth edition.* New York: The Free Press.

SKILTON, Walter, CAMERON, Stuart, SANDERSON, Penelope (1998). Supporting cognitive work analysis with the work domain analysis workbench (WDAW). *Proceedings of the Australasian Computer Human Interaction Conference (OzCHI'98), Adelaide* (pp. 260-267). IEEE. Consulté le 9 février 2009, tiré de IEEE, http://ieeexplore.ieee.org

SOTTET, Jean-Sébastien, CALVARY, Gaelle, FAVRE, Jean-Marie (2005). Ingénierie de l'interaction homme-machine dirigée par les modèles. *Actes des 1ères journées sur l'ingénierie dirigée par les modèles, Paris, France (IDM05)* (pp. 67-82). Conculté le 9 février 2009, tiré de http://planet-mde.org/idm05

TIDWELL, Jenifer (2005). *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly & Associates.

TUFTE, Edward R. (1983). *The visual display of quantitative information.* Cheshire: Graphics Press.

TULLIS, Thomas S. (1997). Screen design. In M. G. Helander, T. K. Landauer et P. V. Prabhu (éd.), *Handbook of Human-Computer Interaction, second edition.* Amsterdam: North Holland.

TVERSKY, Amos, KAHNEMAN, Daniel (1981). The framing of decisions and the psychology of choice. *Science*, 211(4481), 453-458.

VESSEY, Iris (1991). Cognitive fit: a theory-based analysis of the graphs versus tables literature. *Decision Sciences*, 22(2), 219-240.

VICENTE, Kim J. (1999a). *Cognitive work analysis: toward safe, productive, and healthy computer-based work.* Mahwah: Laurence Erlbaum Associates.

VICENTE, Kim J. (1999b). Wanted: psychologically relevant, device- and event-independent work analysis techniques. *Interacting with Computers*, 11, 237-254.

VICENTE, Kim J. (2002). Ecological Interface Design: Progress and challenges. *Human Factors*, 44, 62-78.

VICENTE, Kim J., CHRISTOFFERSEN, Klaus, PEREKLITA, Alex (1995). Supporting Operator Problem Solving Through Ecological Interface Design. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4). 529-545.

VICENTE, Kim J., RASMUSSEN, Jens (1992). Ecological interface design: theoretical foundations. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(4), 589-606.

WARE, Colin (2004). *Information visualization: perception for design.* San Francisco: Morgan Kaufmann Publishers.